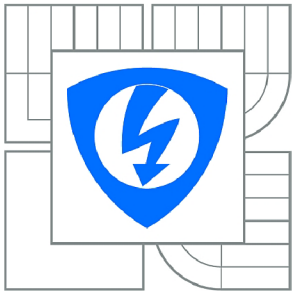
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

LABORATORNÍ ÚLOHY PRO MIKROKONTROLÉR ARM CORTEX-M3.

LABORATORY ASSIGNMENT FOR ARM CORTEX-M3 MICROCONTROLLER

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

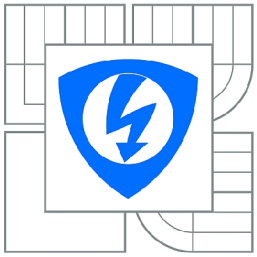
AUTOR PRÁCE — RADEK LIBICHER

AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ MACHO, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Radek Libicher	***ID:***	134542
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Laboratorní úlohy pro mikrokontrolér ARM Cortex-M3.

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s mikrokontroléry ARM Cortex-M3, vývojovou destičkou mbed NXP LPC1768 a příslušnými vývojovými nástroji.
2. Navrhněte vzorové laboratorní úlohy demonstrující využití periferních subsystémů mikrokontrolérů ARM Cortex-M3.
3. Navrhněte rozšiřující obvody pro připojení klávesnice, LCD displeje, sedmisegmentového displeje, tlačítek, LED diod, snímače teploty atd.
4. Navržené úlohy realizujte. Pro každou úlohu vytvořte zadání a návod. Vytvořte vzorový program v jazyce C/C++ a odlaďte jej.

DOPORUČENÁ LITERATURA:

Váňa, Vladimír. ARM pro začátečníky. BEN - technická literatura, 2009. ISBN 978-80-7300-246-6.

Termín zadání:	11.2.2013	***Termín odevzdání:***	27.5.2013

Vedoucí práce: Ing. Tomáš Macho, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato bakalářská práce obsahuje návrh rozšiřujících obvodů a tři laboratorní úlohy pro vývojový kit MBED NXP LPC1768. Dokument je rozdělen na tři základní části. První část se zabývá popisem mikroprocesorů ARM, vývojové desky a vývojových nástrojů. V druhé části bylo navrženo šest rozšiřujících obvodů. Z těchto obvodů byla navržena i realizována Deska periferií jako jednostranný plošný spoj. Pomocí této desky je možný snadný vývoj aplikací. Dále byl k obvodům vytvořen soubor knihoven v jazyce C++ pro jejich snadné a efektivní ovládání. V poslední části se nachází tří laboratorní úlohy – Digitální hodiny využívající hodiny reálného času (RTC), Měření vstupního napětí pomocí AD převodníku, Digitální teploměr s historií naměřených dat. U každé z těchto úloh byl vytvořen detailní popis řešení úlohy včetně vývojového diagramu. Tyto úlohy jsou koncipovány pro výuku mikroprocesorové techniky na elektrotechnických fakultách.

Klíčová slova

mikrokontrolér, vývojový kit, MBED NXP LPC1768, ARM, laboratorní úlohy, uVision, digitální hodiny, digitální teploměr, A/D převodník

Abstract

This bachelor's thesis contains expansion circuits and three laboratory assignments for MBED NXP LPC1768 development kit. The document is divided into three parts. The first part deals with the description of the ARM microprocessor, development board and development tools. In the second part, six expansion circuits were designed. These circuits have been used to create Peripheral Board in the form of one-sided PCB. Using this board, you can easily develop applications. A set of C++ libraries was created for easy and efficient operation. The last part is dedicated to three laboratory assignments – Digital Clock Using Real-time Clock, Input Voltage Measurement with the AD Converter, Digital Thermometer with a History of Measured Data. Detailed descriptions of the solutions for each of these assignments were created, including the flowchart. These assignments are conceived for the reasons of microprocessor technology teaching at the faculties of electrical engineering.

Keywords

microcontroller , development kit, MBED NXP LPC1768, ARM, laboratory assignments, uVision, digital clock, digital thermometer, A/D converter

Bibliografická citace

LIBICHER, R. *Laboratorní úlohy pro mikrokontrolér ARM Cortex-M3.*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 55 s. Vedoucí bakalářské práce Ing. Tomáš Macho, Ph.D..

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Laboratorní úlohy pro mikrokontrolér ARM Cortex-M3 jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 27. května 2013 .....

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Tomáši Machovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: 27. května 2013 .....

podpis autora

1 OBSAH

1 Obsah 7
2 Seznam obrázků a tabulek 8
3 Úvod 9
4 Vlastnosti vývojové desky mbed NXP LPC 1768 10
4.1 Popis vývojové desky 10
4.1.1 ARM architektura 10
4.1.2 Základní popis vývojové desky 10
4.1.3 Periferie desky 11
4.2 Vývojové nástroje 12
4.2.1 Mbed on-line Compiler 12
4.2.2 Keil uVision4 12
5 Návrh rozšiřujících obvodů 14
5.1 Základní koncepce navržených obvodů 14
5.2 Připojení maticové klávesnice 15
5.3 Připojení LCD displeje 16
5.4 Připojení sedmi-segmentového displeje 18
5.5 Připojení tlačítek 20
5.6 Připojení LED diod 21
5.7 Připojení snímače teploty 22
5.8 Deska periferií 24
5.8.1 Napájecí obvod 25
5.8.2 Připojení USB a Ethernet 25
5.8.3 Návrh desky plošného spoje 26
6 Laboratorní úlohy 27
6.1 Digitální hodiny využívající obvod reálného času (RTC) 28
6.1.1 Zadání 28
6.1.2 Teoretický úvod 28
6.1.3 Popis řešení 30
6.2 Měření vstupního napětí pomocí AD převodníku 33
6.2.1 Zadání 33
6.2.2 Teoretický úvod 33
6.2.3 Popis řešení 35
6.3 Digitální teploměr s historií naměřených dat 38
6.3.1 Zadání 38
6.3.2 Teoretický úvod 38
6.3.3 Popis řešení 40
7 Závěr 44
8 Seznam literatury 45
9 Seznam příloh 46

2 SEZNAM OBRÁZKŮ A TABULEK

Obr. 1 Základní popis vývojové desky mbed NPX LPC1768 (převzato z [4]) 11
Obr. 2 Zapojení periferií desky mbed NXP LPC1768 (převzato z [4]) 12
Obr. 3 Prostředí uVision4 13
Obr. 4 Připojení SMD součástky na nepájivé pole pomocí přípravku 14
Obr. 5 Schéma připojení maticové klávesnice 15
Obr. 6 Průběh komunikace řadiče EDE1144 s nadřazeným systémem (převzato a upraveno z [10]) 16
Obr. 7 Schéma připojení LCD displeje 16
Obr. 8 Komunikace s posuvným registrem 4096 (převzato z [9]) 18
Obr. 9 Znaková sada paměti CGRAM řadiče HD4470 (převzato z [11]) 18
Obr. 10 Schéma připojení sedmi-segmentového displeje 20
Obr. 11 Schéma připojení tlačítek 21
Obr. 12 Schéma připojení LED diod 21
Obr. 13 Formát vstupních dat pro řízení jasu LED diod pomocí obvodu TLC5940 (převzato z [8]) 22
Obr. 14 Schéma připojení snímače teploty 23
Obr. 15 Struktura vnitřních registrů obvodu LM92 (převzato z [7]) 24
Obr. 16 Náhled na Desku periferií 25
Obr. 17 Napájecí obvod Desky periferií 25
Obr. 18 Připojení USB a Ethernet 26
Obr. 19 Stavový diagram digitálních hodin 30
Obr. 20 Vývojový diagram – Digitální hodiny využívající obvod reálného času 32
Obr. 21 Časový diagram zápisu do obvodu TLC5940 (převzato a upraveno z [8]) 35
Obr. 22 Vývojový diagram – Měření vstupního napětí pomocí AD převodníku 37
Obr. 23 Popis ovládaní programu Temperature Monitor 40
Obr. 24 Popis Temterature Register (převzato z [7]) 40
Obr. 25 Vývojový diagram – Digitální teploměr s historií naměřených dat. 43

Tab. 1 Základní rysy vývojové desky mbed NXP LPC1768 11
Tab. 2 Seznam součástek pro připojení maticové klávesnice 15
Tab. 3 Seznam součástek pro připojení LCD displeje 16
Tab. 4 Seznam součástek pro připojení sedmi-segmentového displeje 20
Tab. 5 Seznam součástek pro připojení tlačítek 21
Tab. 6 Seznam součástek pro připojení LED diod 22
Tab. 7 Volba slave adresy obvodu LM92 23
Tab. 8 Seznam součástek pro připojení snímače teploty 23
Tab. 9 Popis připojení pinů obvodu TLC5940 na Desku periferií 34
Tab. 10 Popis speciálních znaků pro metodu printf 48

3 ÚVOD

V poslední době se stále častěji můžeme setkat se zařízeními obsahujícími mikrokontrolér. Tento nárůst je způsoben zejména jejich nízkou cenou a širokým sortimentem. Jejich výkon a počet periferií se neustále zvyšuje. Jednoduché aplikace tak mohou obsahovat jeden levný mikrokontrolér pro řízení celého zařízení. Velký rozvoj v této oblasti započal v 80. letech minulého století, kdy se staly velmi populární mikrokontroléry Intel 8080 či 8051. Pro ně bylo typické programování na nízké úrovni nejčastěji v jazyce symbolických adres. Postupným vývojem se stal dominantní mezi programovacími nástroji jazyk C. Dnes se můžeme setkat i s jazykem C++ či jinými objektově orientovanými jazyky, které se hojně používají pro vývoj mikroprocesorových systémů.

Tato práce obsahuje zadání tří laboratorních úloh určené pro studenty elektrotechnických fakult pro osvojení poznatků z mikroprocesorové techniky a programovacích dovedností. Tyto úlohy jsou určeny pro vývojovou desku mbed NXP LPC1768, která obsahuje dostatečně výkonný 32bitový mikrokontrolér pro vývoj aplikací v jazyce C++. Tato deska však postrádá jakékoliv vstupně/výstupní zařízení pro komunikaci s uživatelem, kterými mohou být například LCD displeje, klávesnice, signalizační LED diody a mnoho dalších. Z těchto důvodů se kapitola 4 věnuje návrhu rozšiřujících obvodů pro připojení k vývojové desce. Konstrukci navržených obvodů mohou studenti provést sami s využitím nepájivého pole přímo v laboratořích. Pokud se studenti nechtějí zdržovat konstrukcí obvodů na nepájivém poli, můžou využit Desku periferií, která byla k těmto účelům speciálně navržena a je popsána na konci kapitoly 4. Tím je umožněno odzkoušení aplikace přímo na reálném zařízení. Aby vývoj programů nebyl příliš časově náročný, byly pro některé rozšiřující obvody vytvořeny knihovny, které umožňují jejich snadné ovládání. Studenti by tak měli být schopni navrhnout strukturu programu a provést realizaci programu i s odzkoušením na příslušném zapojení. Popisem těchto knihoven je uveden v příloze.

Pro správné vytvoření laboratorních úloh je důležité, aby byla předkládaná látka dostatečně srozumitelná pro studenty. Toto je však dosti obtížné a do velké míry závisí na zkušenostech a kvalitách vyučujících, ale také na vybavenosti laboratoří či časovým rozsahem výuky. Všechny vytvořené úlohy jsou sestavené tak, aby byly zvládnutelné v rámci 3 vyučovacích hodin (150 min). Obtížnost jednotlivých úloh byla nastavena tak, aby méně zdatné studenty náročnost neodradila a studenti v programování zběhlí se nenudili. Mezi další nedílné části úspěšného předání znalostí studentům pomocí laboratorních úloh je jistá zábavnost či zajímavost úloh. Proto byl při vytvoření úloh kladen důraz na jejich originalitu a možnost praktického využití.

4 VLASTNOSTI VÝVOJOVÉ DESKY MBED NXP LPC 1768

4.1 Popis vývojové desky

4.1.1 ARM architektura

ARM je rodina 32bitových mikroprocesorů navržených dle RISC koncepce. Jejich výrazný rozmach započal v 90. letech minulého století a dnes již patří mezi nejrozšířenější na světě v počtu vyrobených kusů. Nacházejí uplatnění v mnoha aplikacích spotřební elektroniky od kalkulátorů, mobilních telefonů až po výkonné herní konzole či tablet PC. Díky svým vlastnostem se také hodně využívají ve vestavěných systémech. Přístup ARM procesorů je v integraci co možná nejméně tranzistorů oproti konkurenci a tím dosažení nízké spotřeby. Z tohoto důvodu jsou tyto procesory často nasazeny v systémech napájených bateriemi.

Základní charakteristiku procesorů ARM můžeme shrnout do několika následujících bodů.

- 32bitová architektura i 32bitová datová sběrnice
- Load/store architektura – Pro provedení aritmeticko-logické operace musí být operandy nejdříve načteny do registrů. Po provedení operace je možné výsledek nahrát zpět do paměti. Většina instrukcí tak pracuje pouze s registry.
- Soubor šestnácti 32bitových registrů pro všechny procesory ARM
- Dva adresové módy – Použití čítače instrukcí případně bázové adresy v registru.
- Dvě úrovně priority přerušení.
- Výkonný a efektivní instrukční soubor koncipovaný pro kompilátory vyšších programovacích jazyků

4.1.2 Základní popis vývojové desky

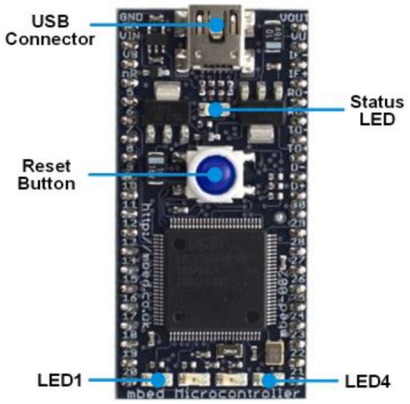
Tato vývojová deska je navržena pro připojení do nepájivého pole se 40 vývody odpovídající rozložení pinů pouzdra DIP. Obsahuje zabudované USB rozhraní pro programování, které se po připojení k PC chová jako klasický flash disk. Pro nahrání nového programu je nutné překopírování zkompilovaného programu ve formátu .bin a následné resetování zařízení. Tím dojde ke zkopírování programu do interní paměti a k jeho spuštění. Tímto řešením můžeme s vývojovou deskou pracovat na operačních systémech Windows, MAC OS či Linux bez nutnosti instalace speciálních ovladačů. Pro vývoj programů je k dispozici zdarma integrovaný on-line kompilátor přímo na

webu výrobce. K efektivnějšímu vývoji software je však výhodnější využít vývojová prostředí jako jsou Keil uVision nebo Code Red.

Deska je založena na ARM Cortex-M3 jádře běžící na 96 MHz. Obsahuje dále 512 KB Flash, 32 KB RAM a velké množství rozšiřujících periferií. Vše je shrnuto v Tab. 1. Zde je patrné, že deska nabízí velké množství periferií, jenž jsou vyvedeny na výstup desky. Dodejme, že piny P5 až P30 lze použít i jako digitální vstup či výstup. Obr. 2 znázorňuje základní popis vývojové desky.

Jádro	Frekvence	FLASH	RAM	Odběr	Periferie
ARM Cortex-M3	96 MHz	512 KB	32 KB	60-120 mA	Ethernet, USB, SPI, I2C, UART, ADC, DAC, PWM

Tab. 1 Základní rysy vývojové desky mbed NXP LPC1768

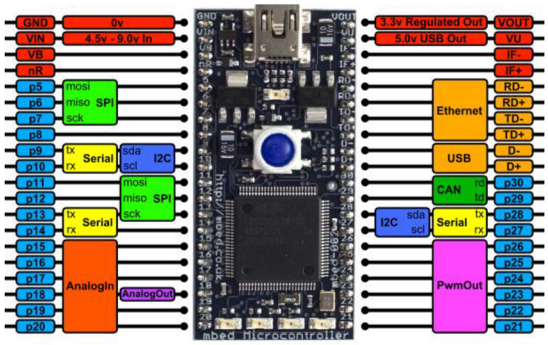


Obr. 1 Základní popis vývojové desky mbed NPX LPC1768 (převzato z [4])

4.1.3 Periferie desky

Jak již bylo výše uvedeno, deska poskytuje velké množství periferních obvodů. Tyto obvody je možné rozdělit do několika kategorií. Jako první zmíníme sériová periferní rozhraní, kterých deska MBED má velké množství. Mezi ně patří SPI, I2C a UART. Přes uvedená rozhraní lze velmi efektivně komunikovat s ostatními integrovanými obvody, kterými mohou být EEPROM paměti, externí A/D či D/A převodníky, displeje a mnoho dalších obvodů. Použití těchto rozhraní lze velmi efektivně zjednodušit návrh daného zařízení, namísto užití paralelní komunikace. Mezi další skupinu patří obvody pro styk s analogovými obvody. K těmto účelům se používají AD či DA převodníky. AD převodník na desce pracuje na principu postupné aproximace s rozlišením 12 bitů a s maximální vzorkovací periodou 200 kHz. Podrobnější popis převodníku se nachází v kapitole 6.2.2.1, v níž byl převodník použit v rámci řešení úlohy. Pro ovládání analogových zařízení je na desce integrován 12bitový DA převodník, na kterém je možné nastavit výstupní napětí v rozsahu 0 až 3,3 V. Jak je zřejmé z Obr. 2 převodník má pouze jeden kanál vyvedený na pin p18. Do této skupiny lze zařadit i obvod pro generování pulzně šířkové modulace (PWM), jenž

v mnohých aplikací může nahradit DA převodník. Zejména pokud je potřeba například zajistit regulaci pohonů. Do poslední kategorie patří rozhraní převážně používaná pro komunikaci s nadřazeným systémem, kterými jsou USB a Ethernet. Dále je nezbytné poznamenat, že výrobce desky na webu [4] nabízí ke stažení knihovny pro snadné ovládání všech výše uvedených periferií v jazyce C++, což výrazně zjednodušuje jejich obsluhu.



Obr. 2 Zapojení periferií desky mbed NXP LPC1768 (převzato z [4])

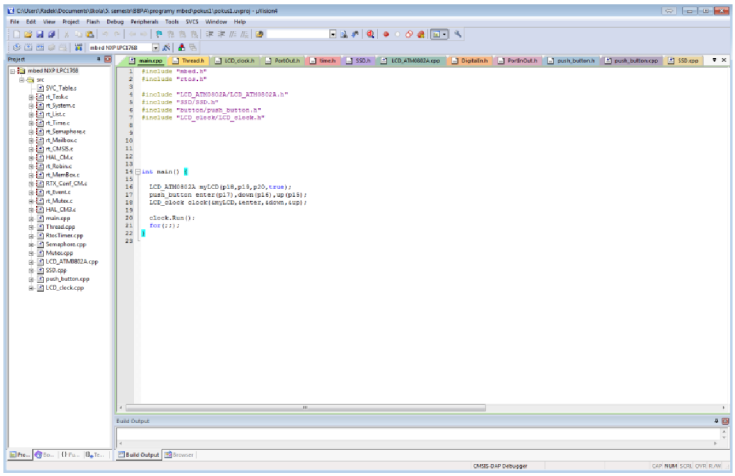
4.2 Vývojové nástroje

4.2.1 Mbed on-line Compiler

Mbed Compiler poskytuje možnost jednoduchého vytváření aplikací v C/C++, který je koncipován tak, aby bylo možné programy vytvářet přímo na webu a poté je jen stáhnout do vývojové desky. Není nezbytné instalovat jakýkoliv software. S touto webovou aplikací tedy můžete pracovat na libovolném operačním systému, který obsahuje webový prohlížeč a možnost připojení k internetu. Kompilátor nabízí formátování kódu či generování automatické dokumentace. Postrádá však možnost jakéhokoliv ladění programu pomocí simulátoru.

4.2.2 Keil uVision4

uVision 4 poskytuje tvorbu projektů, editaci zdrojových kódu, kompilaci, ladění přímo na čipu či kompletní simulaci v jediném vývojovém prostředí. Tento vývojový nástroj využívá plné možnosti mikrokontrolérů ARM a tím umožňuje rychlý a efektivní vývoj aplikací. Pomocí jednoduchého průvodce můžeme vytvořit nový projekt pro ARM procesor v jazyce C/C++ případně použít import i z výše uvedeného on-line kompilátoru. Na Obr. 3 zobrazeno základní rozhraní prostředí uVision pro tvorbu projektu. V jeho levé části se nachází soubor knihoven, které je zde nutné před použitím v projektu vložit. Pravá část obsahuje vybraný zdrojový kód pro editaci. V horní části si můžeme zvolit ladící nástroj.



Obr. 3 Prostředí uVision4

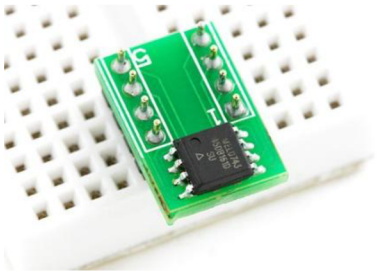
5 NÁVRH ROZŠIŘUJÍCÍCH OBVODŮ

5.1 Základní koncepce navržených obvodů

Mikrokontrolér ARM LPC 1768 obsahuje celkem čtyři 32bitové porty, přičemž pouze 25 V/V pinů je vyvedeno na výstup vývojové desky pro připojení rozšiřujících obvodů. Většina těchto vývodů má alternativní funkci pro periferie mikrokontrolérů jako jsou sériová rozhraní, A/D převodník atd.

Jednotlivé typu navržených rozšiřujících obvodů odpovídají požadavkům bodu číslo 3 zadání této práce, kde je striktně vymezeno jaké typy obvodů mají být navrženy. Samotné připojení těchto obvodů vychází z omezeného počtu V/V pinů desky MBED. Z těchto důvodů byla ve velké míře navržena sériová komunikace s rozšiřujícími obvody.

Tato kapitola ukazuje zapojení šesti rozšiřujících obvodů pro desku MBED. Na konci této kapitoly je popsána Deska periferií, která všechny obvody sdružuje do jedné rozšiřující desky. Tato Deska periferií byla i realizovaná a odzkoušena. Veškerá dokumentace včetně celkového schéma se nachází v příloze. S využitím této desky je vývoj laboratorních úloh velmi usnadněn. Všechny následující zapojení však lze realizovat i na nepájivém poli. Jednotlivé obvody jsou navrženy i s ohledem na jejich snadnou realizovatelnost na desce nepájivého pole, čemuž je uzpůsobena i deska MBED. Proto jsou použity integrované obvody výhradně s pouzdry DIL případně jim podobné. Pro obvody s pouzdry na SMD montáž je nezbytné použití odpovídajícího přípravku. Na Obr. 4 je ukázáno připojení SMD obvodu na nepájivé pole s využitím přípravku s konverzí na rozložení vývodu odpovídající pouzdru DIL.

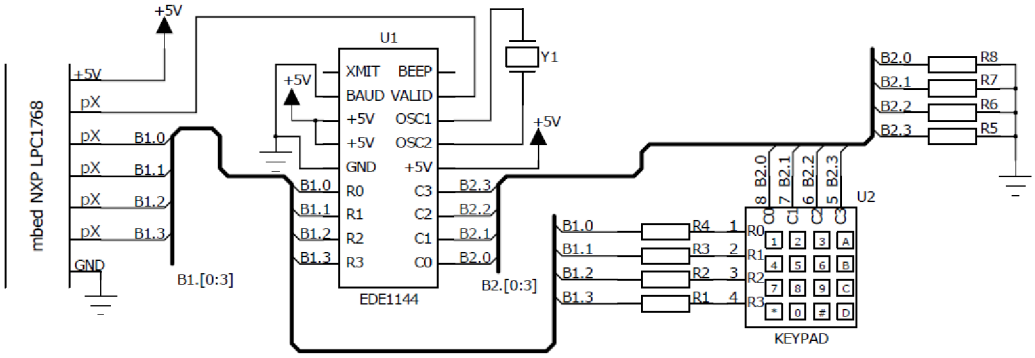


Obr. 4 Připojení SMD součástky na nepájivé pole pomocí přípravku

Vývojová deska nabízí k napájení rozšiřujících obvodů napěťové úrovně 3,3V a 5V. I/O piny desky však používají logické úrovní o napětí 3,3V, ale napětí 5V na vstupním pinu je přípustné bez poškození obvodu. Maximální dovolený proud jednoho pinu je 4 mA, celé desky pak 400 mA. Při návrhu byly použity IO logiky TTL i CMOS. Pro napájení obvodů TTL byla použita větev 5V, naopak pro CMOS větev 3,3V. Napětí 3,3V je na vstupních pinech obvodů TTL dostačující pro zajištění úrovně H. Tímto byla vyřešena napěťová kompatibilita logických úrovní rozšiřujících obvodů, bez nutnosti připojení převodníku napěťových úrovní.

5.2 Připojení maticové klávesnice

Klávesnice slouží jako vstupní zařízení pro uživatele u mnoha aplikací. Jde o množinu spínacích prvků uspořádaných tak, aby ovládání daného zařízení bylo jednoduché a přehledné.



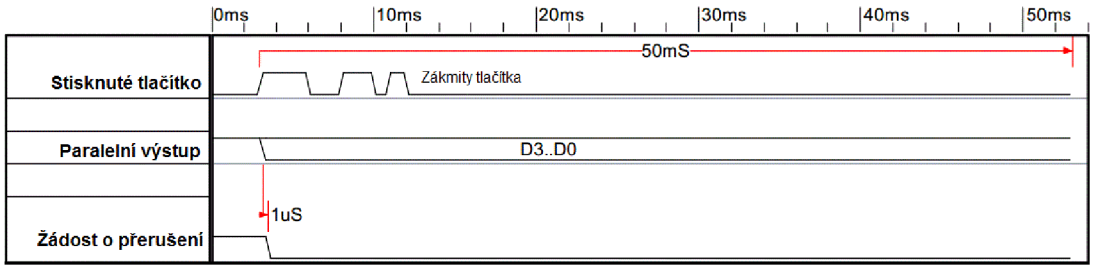
Obr. 5 Schéma připojení maticové klávesnice

Označení	Typ součástky (hodnota)	Počet kusů
U1	EDE1144	1
U2	KEYPAD 4x4	1
Y1	QM 4 MHz	1
R1-4	RR 330R	4
R5-8	RR 4K7	4

Tab. 2 Seznam součástek pro připojení maticové klávesnice

Obr. 5 ukazuje zapojení maticové klávesnice s využitím obvodu EDE1144. Tento řadič zajišťuje dynamické čtení jednotlivých kláves a komunikaci s nadřazeným systémem, kterou je možno zabezpečit pomocí sériové linky UART případně paralelním připojením ve formě BCD kódu. Pro obě tyto možnosti lze také aplikovat přerušovací subsystém, který se aktivuje při každém stisku libovolné klávesy. Navržené zapojení využívá jednoduché paralelní připojení s přerušením. Hodnoty rezistorů R1-4 i R5-8 byly zvoleny podle doporučení výrobce IO EDE1144. Rovněž jako zdroj hodinového signálu byla použita doporučená hodnota krystalu Y1 4MHz. Výstupní piny XMIT a BEEP slouží k sériovému výstupu UART a také k zvukové signalizaci aktivního tlačítka. Pro dané zapojení tedy nejsou podstatné.

Celé výše uvedené zapojení pracuje následovně. V klidovém stavu je na všech výstupech R0 až R3 logická úroveň 1. Při stisku kterékoliv klávesy dojde ke změně logické úrovně na příslušném vstupu Cx, což řadič vyhodnotí a začne postupným přepínáním výstupů Rx snímat tlačenou klávesu. Po nalezení dané klávesy je možno po dobu 50 ms hodnotu stisknuté klávesy načíst na pinech R0 až R3 v podobě BCD kódu. Po uplynutí 1us je taktéž vyslán požadavek na přerušení na výstupu VALID. Celý průběh čtení a komunikace je zobrazen na Obr. 6. Řadič je schopný potlačit případné zákmity tlačítek, proto je není nutné ošetřit externě.

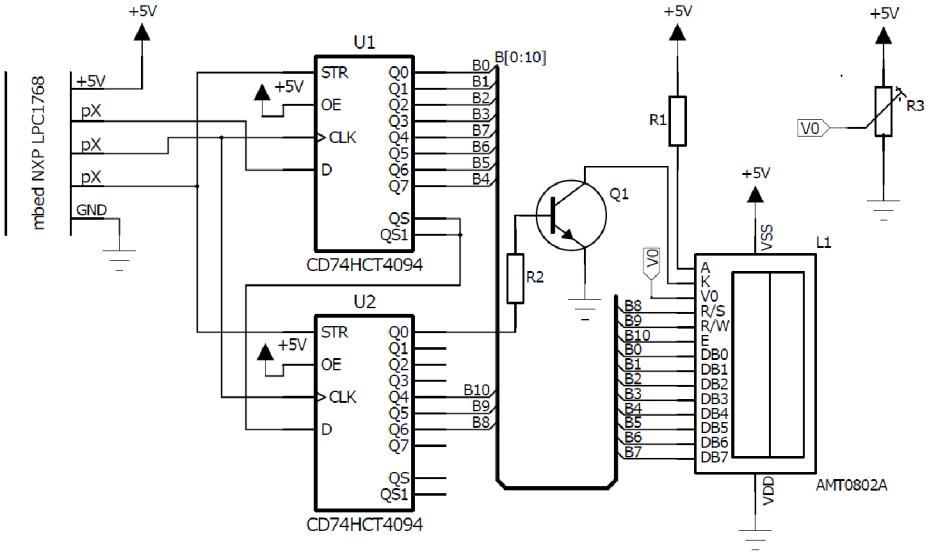


Obr. 6 Průběh komunikace řadiče EDE1144 s nadřazeným systémem (převzato a upraveno z [10])

5.3 Připojení LCD displeje

LCD displej patří mezi základní zobrazovací zařízení pro vizualizaci stavu zařízení. Můžeme je najít v mnoha zařízeních spotřební elektroniky. Lze je rozdělit podle mnoha kritériích. Jedno ze základních dělení může být na alfanumerické a grafické. V prvním z nich probíhá zobrazování pomocí jednotlivých znaků uložených v paměti řadiče. U grafických displejů probíhá vykreslování po jednotlivých bodech. První varianta bude pro naše účely dostačující.

Pro připojení k vývojové desce byl vybrán alfanumerický LCD displej AMT0802A s integrovaným řadičem HD44780. Zapojení je znázorněno na Obr. 7.



Obr. 7 Schéma připojení LCD displeje

Označení	Typ součástky (hodnota)	Počet kusů
U1-2	CD74HCT4094	2
L1	LCD ATM0802A	1
Q1	BC547A	1
R1	RR 10R	1
R2	RR 1K5	1
R3	RR 10K	1

Tab. 3 Seznam součástek pro připojení LCD displeje

Pro snížení počtu pinu nutných pro komunikaci s řadičem bylo použito zapojení přes dvojici sériově zapojených posuvných registrů typu 4094. Tímto způsobem můžeme

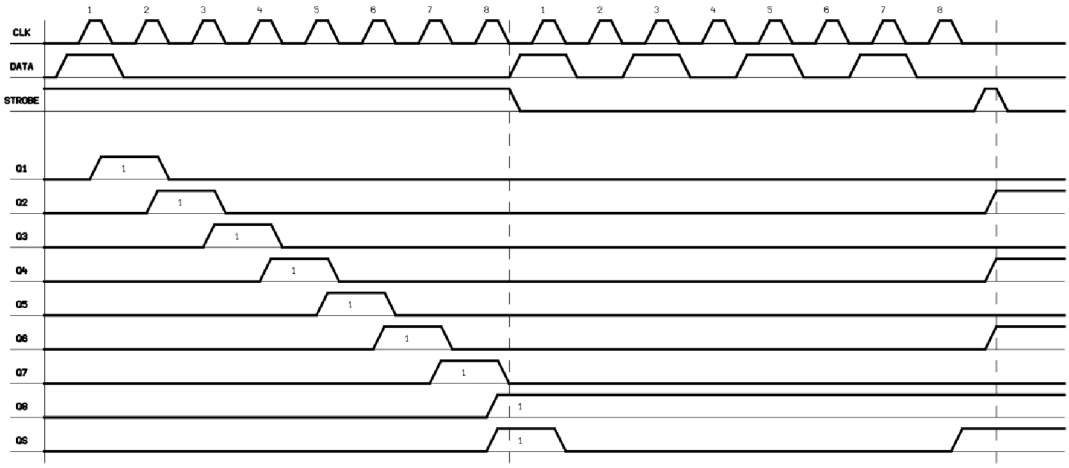
komunikovat s řadičem pouze s prostřednictvím tří signálových vodičů. Je však nutné dodat, že tím znemožníme zpětné načítání dat z řadiče do nadřazeného systému, jelikož tento typ posuvných registrů umožňuje jen jednosměrnou komunikaci. Tento fakt ovšem není ve většině aplikací omezující. Vstupní vývody D, CLK a STR slouží pro vstup sériových dat. Vstup OE je použit, pokud chce s nějakého důvodu odpojit výstupní obvody od okolí (stav vysoké impedance). Pro naše účely jej tedy můžeme trvale připojit na úroveň H. Řadič HD44780 nabízí možnost volby jak 8bitové tak 4bitové komunikace, jelikož by 4bitová komunikace byla velmi komplikovaná a pomalá, bylo zvoleno 8bitové zapojení. Změnou napětí na vstupu V0 můžeme řídit kontrast displeje. K tomu slouží trimr R3, jehož hodnota 10K byla zvolena dle doporučení katalogového listu výrobce. Pro ovládání podsvícení displeje byl použit tranzistor BC547A ve spínacím režimu s vnuceným proudovým zesilovacím činitelem β_v=20. Jednotlivé hodnoty použitých rezistorů R1 a R2 byly zvoleny pro doporučený proud posvícením 60mA a vypočteny podle:

$I_F = 60\ mA - požadovaný\ proud\ pro\ posvícení$
$U_{CC} = 5\ V - napájecí\ napětí$
$U_{LED} = 4{,}1\ V - úbytek\ napětí\ na\ podsvícení\ dipleje$
$U_{BE} = 0{,}6\ V - úbytek\ napětí\ mezi\ bází\ a\ emitorem$
$U_{CE} = 0{,}3\ V - úbytek\ napětí\ mezi\ CE\ při\ sepnutém\ stavu\ při\ I_C = 60\ mA\ a\ I_B = 3\ mA$
$\beta_v = 20 - vnucený\ proudový\ zesilovací$ činitel

$$R_1 = \frac{U_{CC} - U_{CE} - U_{LED}}{I_F} = \frac{5 - 0{,}3 - 4{,}1}{0{,}06} = \mathbf{10\Omega} \quad (1)$$

$$R_2 = \frac{Ucc - U_{BE}}{\frac{I_F}{\beta_v}} = \frac{5 - 0{,}6}{\frac{0{,}06}{20}} = 1466\Omega \approx \mathbf{1{,}5k\Omega} \quad (2)$$

Komunikaci s nadřazeným systémem můžeme rozdělit na dvě části. Jde o sérioparalelní převod pomocí výše zmíněných posuvných registrů a vlastní komunikaci s řadičem HD4470. Jak již bylo výše uvedeno, pro vkládání dat postačí tři signály a to Strobe (STR), Clock(CLK) a Data (D). Komunikace probíhá tak, že je MSB bit vysílaného slova vystaven na vývod D a poté je generován kladný pulz CLK. Stejný postup platí i pro následující bity. Chceme-li vidět ihned odezvu na výstupu, ponecháme signál STROBE v úrovni H. Pro naše účely, však potřebujeme ryzí sériově/paralelní převodník, proto musíme vyslat kladných impulz STROBE až po přenesení celého slova. Odeslání dvou 8mi bitových slov v těchto různých režimech je znázorněno na Obr. 8.



Obr. 8 Komunikace s posuvným registrem 4096 (převzato z [9])

Řadič LCD displeje obsahuje dvojici paměťových bloků. V paměti DDRAM jsou uloženy znaky, které se mají zobrazit na displeji. Zatímco zápisem do paměti CGRAM můžeme vytvořit nové znaky, jenž se nenachází ve standardní znakové sadě. Komunikace probíhá zasíláním jednotlivých instrukcí a dat do řadiče. Připomeňme, že zpětné čtení z pamětí DDRAM a CGRAM není možné. Dále je nutné dodat, že po připnutí na DDRAM paměť je nutné zvolit 7bitovou adresu a tím zvolit pozici kurzoru. Pro vybraný LCD displej začíná první řádek adresou 00h a druhý řádek 40h. Následným zápisem dat bude vybraný znak zobrazen na displeji. Základní znaky odpovídají ASCII kódování. Celá znaková sada je znázorněna v Obr. 9. Před samotným zápisem znaků je nutné provést inicializační sekvenci s nastavením 8mi bitové komunikace.

Obr. 9 Znaková sada paměti CGRAM řadiče HD4470 (převzato z [11])

5.4 Připojení sedmi-segmentového displeje

Mezi další zobrazovací zařízení patří sedmi-segmentové displeje. Jsou určeny zejména k zobrazování číselných údajů, případně jiných jednoduchých znaků. Jejich

postatou je 7 svítivých diod uspořádaných tak, aby bylo možné zobrazit libovolnou číslovku.

Na Obr. 10 můžeme vidět zapojení využívající dvojici sedmi-segmentových displejů. Je zde použito převodníků z BCD kódu na sedmi-segmentový displej. Jelikož jsou všechny výstupy převodníku invertovány je nutné je připojit na katodu jednotlivých zobrazovacích LED diod. Je tedy nezbytné použít displej se společnou anodou. Rezistory R1-R14 je nutné zvolit pro optimální velikost proudu segmenty displeje. Jejich hodnotu zvolíme dle:

$I_F = 20\,mA - zvolený\ proud\ pro\ jeden\ segment\ displeje$
$U_{CC} = 5\,V - napájecí\ napětí$
$U_{LED} = 2\,V - úbytek\ napětí\ na\ segmentu\ displeje\ při\ I_F = 20\,mA$

$$R_{1-14} = \frac{U_{CC} - U_{LED}}{I_F} = \frac{5-2}{0{,}02} = \mathbf{150\ \Omega} \tag{3}$$

Vstup LT převodníku slouží pro rozsvícení všech segmentů při připojení napájecího napětí a tím umožnit jejich funkční kontrolu (tzv. Lamp test). Připojíme-li k tomuto vývodu paralelní RC obvod zapojený pro zemi, proběhne rozsvícení všech segmentů po danou dobu. Tento čas můžeme přibližně vypočítat jako dobu, za kterou se vybije kondenzátor na hodnotu maximálního napětí L úrovně v TTL obvodech. Nejprve vypočteme požadovanou hodnotu časové konstanty RC obvodu pro čas trvání Lamp testu 2 s podle:

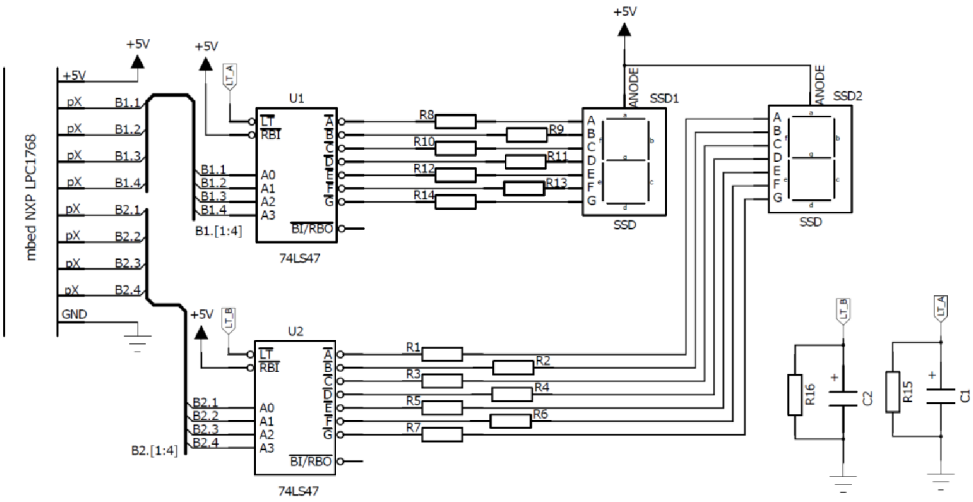
$t = 2\,s - zvolený\ čas\ trvání\ Lamp\ testu$
$U_{LMAX} = 0{,}8\,V - maximální\ napětí\ úrovně\ L\ v\ TTL\ obvodech$

$$U_{Lmax} = U_{cc} e^{-\frac{t}{\tau}} \rightarrow \tau = -\frac{t}{\ln\frac{U_{Lmax}}{U_{cc}}} = -\frac{2}{\ln\frac{0{,}8}{5}} = \mathbf{1{,}091\ s} \tag{4}$$

Nyní zvolíme hodnotu rezistorů R_{15} a R_{16} a dopočítáme kapacitu příslušných kondenzátorů. Tímto docílíme přibližnou dobu trvání Lamp testu 2s.

$R_{15,16} = 10\,K\Omega$

$$C_{1,2} = \frac{\tau}{R_{15,16}} = \frac{1{,}091}{10^4} = 1{,}091 \cdot 10^{-4}\,F \approx \mathbf{100\ \mu F} \tag{5}$$



Obr. 10 Schéma připojení sedmi-segmentového displeje

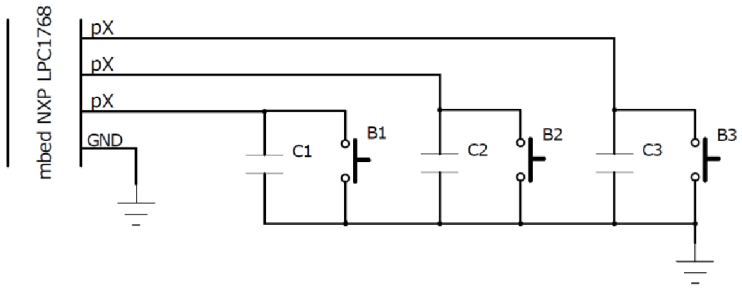
Označení	Typ součástky (hodnota)	Počet kusů
U1-2	74LS47	2
R1-14	RR 150R	14
R15-16	RR 10K	2
C1-2	E100M/16V	2
SSD1-2	SA08-11EWA	2

Tab. 4 Seznam součástek pro připojení sedmi-segmentového displeje

Zobrazení číselných hodnot na displeji provedeme nastavením příslušných logických úrovní na vstupních pinech A0 až A3 v BCD kódu. Obvody 74LS47 neobsahují paměť pro uložení znaku, proto je nezbytné podržet logické úrovně na příslušných vývodech Ax.

5.5 Připojení tlačítek

Trojce tlačítek spíná na Obr. 11 příslušné vstupní piny proti nulovému potenciálu. V nesepnutém stavu jsou tedy ve stavu vysoké impedance, proto je nutné pro jednotlivé vstupní piny aktivovat pull-up rezistory integrované ve vývojové desce. Z tohoto důvodu není nutné je zapojovat externě. Jelikož lze předpokládat, že mechanické tlačítka budou po stisku generovat parazitní zákmity, byly k tlačítkům paralelně připojeny kondenzátory, které částečně zákmity eliminují a také potlačí případné rušení. Kapacita kondenzátorů byla zvolena 100 nF. Případné zvýšením jejich kapacity má za následek efektivnější filtraci zákmitů, ale také delší dobu reakce mikroprocesoru na změnu stavu tlačítka.



Obr. 11 Schéma připojení tlačítek

Označení	Typ součástky (hodnota)	Počet kusů
B1-3	P-B1720B	3
C1-3	CK 100N/25V	3

Tab. 5 Seznam součástek pro připojení tlačítek

5.6 Připojení LED diod

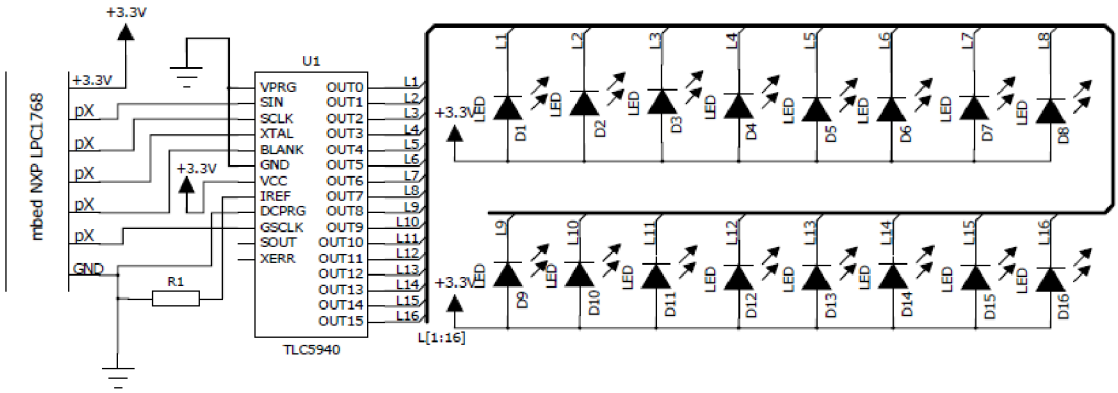
LED diody patří mezi základní indikční součástky. Je možné je připojit přímo na výstupní pin spolu se sériově zapojeným omezovacím rezistorem. Případně při nedostatečném výstupním proudu daného pinu použít proudové budiče. Chceme-li ovšem připojit velké množství LED diod je výhodné použít LED drivery.

Pro připojení diod byl vybrán LED driver TLC5940. Ten disponuje 16 kanály, 4096 kroků PWM modulace a také obsahuje proudové výstupy. Rezistor R1 slouží k nastavení maximálního proudu pro jednotlivé LED. Použité LED diody je možné napájet maximálně proudem 20mA. Rezistor R1 tedy vypočteme podle následujícího vztahu.

$I_{out} = 20\,mA - požadovaný\ proud\ pro\ LED\ diodu$

$$I_{out} = 31{,}5\left(\frac{1{,}24}{R_1}\right) \rightarrow R_1 = \frac{31{,}5 \cdot 1{,}24}{I_{out}} = \frac{31{,}5 \cdot 1{,}24}{20 \cdot 10^{-3}} = 1953\,\Omega \approx \mathbf{2\ K\Omega} \qquad (6)$$

Celé zapojení ukazuje Obr. 12.



Obr. 12 Schéma připojení LED diod

Označení	Typ součástky (hodnota)	Počet kusů
U1	TLC5940	1
D1-16	LED 3MM WHITE 4500/20°	16
R1	RR 2K	1

Tab. 6 Seznam součástek pro připojení LED diod

Obvod TLC5940 umožňuje nastavení výstupního proudu pro jednotlivé výstupy OUT0 až OUT15 zápisem 12bitových dat do náležejících GS[1] registrů. Následující vztah ukazuje nastavení GS registru pro jeden výstup OUTx.

$$jas\ [\%] = \frac{GS}{4095}100 \tag{7}$$

$GS - požadovaná\ úroveň\ jasu\ (GS = 0\ až\ 4095)$

Požadovaná data se do GS registrů zapíší prostředním integrovaného posuvného registru. Pomocí vstupů SIN a SCLK se nejdříve provede zápis 192bitového řetězce dat ve formátu podle Obr. 13 do zmíněného posuvného registru. Jako první se tedy odešle MSB bit pro výstup OUT15 a jako poslední LSB bit určený na výstup OUT0. Poté kladným signálem na vstupu XTAL se vykoná přesun dat do příslušných GS registrů a změna se projeví na výstupech OUT. Aby byla změna na výstupech pozorovatelná, je nutné generovat hodinový signál pro PWM modulaci na vstupu GSCLK. Dále je možné nastavit individuální korekce jasu zápisem 6bitových dat do DC[2] registrů. Pro výše uvedené zapojení však toto nastavení není možné, protože vstup VPROG, který slouží k volbě GS případně DC registru, je trvale připojen na úroveň L. Pro většinu aplikací však tento fakt není omezující. Podrobnější popis obvodu je uveden v 6.2 i s praktickou demonstrací použití.

Obr. 13 Formát vstupních dat pro řízení jasu LED diod pomocí obvodu TLC5940 (převzato z [8])

5.7 Připojení snímače teploty

Teplotní snímače můžeme rozdělit na dvě základní skupiny a to analogová a digitální. Výstupní signál analogových snímačů je nejčastěji ve formě napěťové úrovně, která je pomocí A/D převodníku digitalizována a dále zpracovávána v počítači. Značná nevýhoda těchto snímačů je jejich značná náchylnost na rušení. Proto bylo pro připojení k vývojové desce použito digitální čidlo LM92 se sériovým rozhraním.

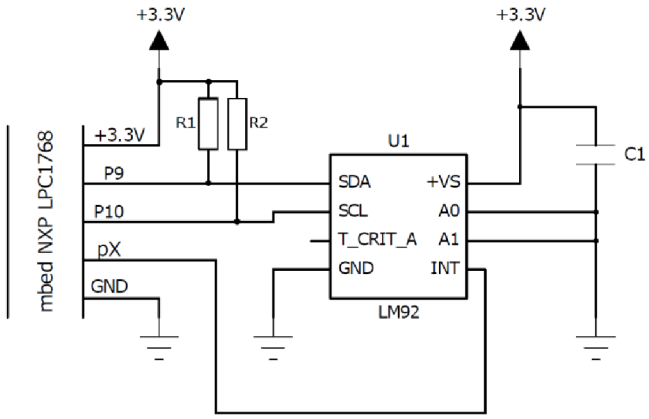
Teplotní čidlo je připojeno k vývojové desce pomocí tří komunikačních vodičů. Signály SDA a SCL jsou určeny pro komutaci přes sběrnici I2C. Výchozí slave adresa obvodu se nastavuje připojením vstupu A0 a A1 na log. 0 případně log. 1. Z Tab. 7 je zřejmé připojením vývodů A0 a A1 můžeme ovlivnit poslední dva bity celé 7bitové adresy. Pro navržené zapojení na Obr. 14 byla nakonfigurována adresa 48h. Vývod INT

[1] Grayscale
[2] Dot correction

slouží k vyvolání přerušení, které je aktivováno po překročení předem nastavené teplotní hladiny v řídících registrech obvodu. Zde je nutné dodat, že pro správnou funkci obvodu je nutné pro vývod INT aktivovat interní pull-up rezistory, jelikož výstup je typu otevřený kolektor. Avšak pro piny SDA a SCL musíme tyto rezistory zapojit externě, protože obvod pro ovládání I2C na desce MBED nenabízí interní připojení těchto rezistorů. Jejich hodnota 2K2 byla zvolena podle doporučení výrobce. Výstup T_CRIT_A, který však není připojen, je možné využít ke generování přerušení v případě překročení nastavené kritické teploty. Pro potlačení rušení z napájení byl připojen filtrační kondenzátor C1. Jeho kapacita byla navržena podle doporučení výrobce.

1	0	0	1	0	A1	A0
MSB						LSB

Tab. 7 Volba slave adresy obvodu LM92

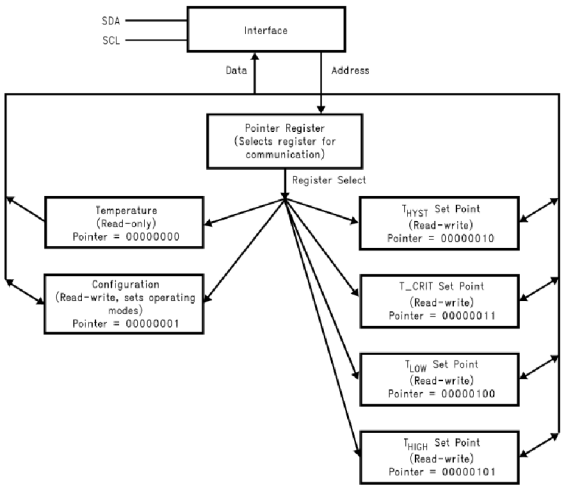


Obr. 14 Schéma připojení snímače teploty

Označení	Typ součástky (hodnota)	Počet kusů
U1	LM92	1
R1-2	RR 2K2	2
C1	CK 100N/63V	1

Tab. 8 Seznam součástek pro připojení snímače teploty

Teplotní číslo LM92 obsahuje 6 speciálních registrů, jejíž struktura je zobrazena na Obr. 15, pro čtení dat a konfiguraci čidla. Čtením registru Temperature získáme 16bitové informace o aktuální teplotě ve formátu dvojkového doplňku, kde LSB bit odpovídá teplotě 0,0625 °C. Obvod umožňuje nastavení dvou teplotních hladin po jejíž překročení, je generován požadavek na přerušení na pinu INT. Volba těchto úrovní se provádí zápisem dat do registrů T_{LOW} a T_{HIGH}. Pro obě úrovně je možné zvolit velikost hystereze v registru T_{HYST}. Případná polarita signálů INT i T_CRIT_A se volí v konfiguračním registru *Configuration*.

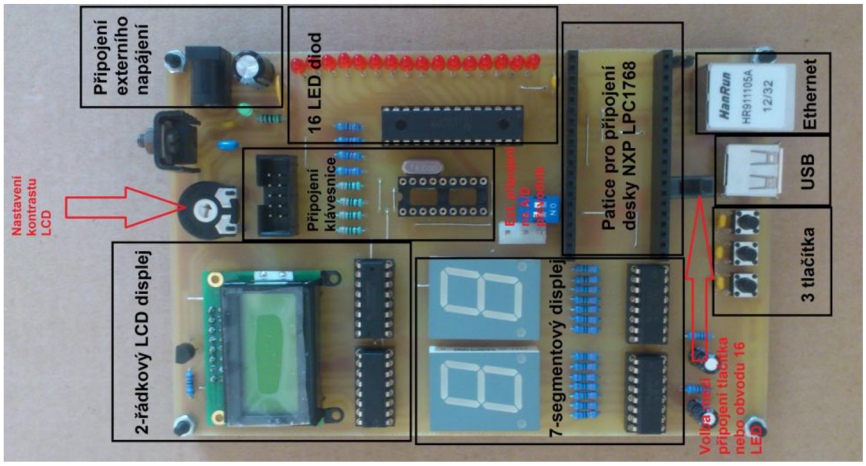


Obr. 15 Struktura vnitřních registrů obvodu LM92 (převzato z [7])

5.8 Deska periferií

Deska periferií byla navržena pro snadnou demonstraci laboratorních úloh bez nutnosti sestavovat obvody na nepájivém poli. Tato deska obsahuje všechny výše uvedené rozšiřující obvody, pro které byly použity všechny V/V piny desky MBED. Dále nabízí připojení přes USB a Ethernet pro komunikaci s okolím.

Některé rozšiřující obvody mají spolu sdílené vývody, jelikož deska MBED měla nedostatečný počet V/V pinů. Přepínání mezi obvody je možné pomocí DIP přepínačů. Tyto přepínače je možné zaměnit za přepojování pomocí jumperů. Na desce je vyvedena trojce konektorů pro připojení externích zařízení. První konektor CON1 slouží k připojení externí maticové klávesnice. Klávesnice byla řešena jako externí zařízení z důvodu úspory místa na desce. Druhý konektor PAD1 je navržen pro snímání externích signálů. A to zejména signálů analogových, protože právě na tyto V/V piny má deska MBED připojeny multiplexované vstupy integrovaného AD převodníku. Poslední hlavní 40ti pinový konektor IC1 je určen k připojení desky MBED s řídícím mikrokontrolérem. Tento konektor tvoří dvě 20ti pinové dutinkové lišty zapojené vedle sebe. Celkový náhled na vyhotovenou desku je uveden na Obr. 16.

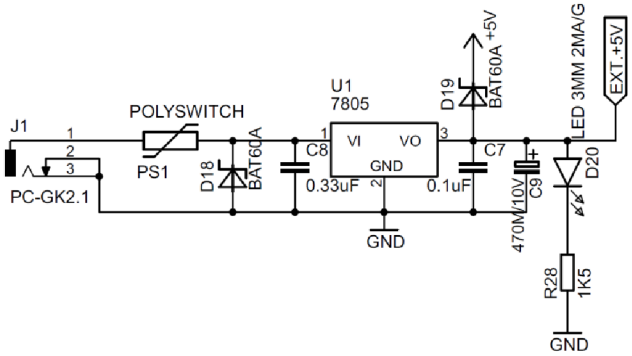


Obr. 16 Náhled na Desku periferií

5.8.1 Napájecí obvod

Napájení všech rozšiřujících obvodů je zajištěno z obvodu, který umožňuje napájení jak z externího zdroje 7,5-12V, tak i z USB, které slouží pro programování. Napájení ze zdroje o napětí více než 12V se nedoporučuje. Mohlo by dojít k přehřátí stabilizátoru. Při napájení pouze z USB však nelze používat obvod se 16ti LED diodami, jelikož by jejich proud překračoval limit určený pro napájení z USB. Při používání tohoto obvodu je tedy nezbytné připojení externího zdroje.

Základem obvodu je lineární stabilizátor 7805, který má dle doporučení připojené odrušovací a filtrační kondenzátory C7-9. Zařízení je chráněno vratnou pojistkou PS1 typu Polyswitch. Pro ochranu proti případnému přepólování slouží dioda D18. Jelikož je možné celé zařízení napájet ze dvou míst (externí zdroj a USB) jsou tyto zdroje k sobě připojeny přes diody BAT60A, které zabraňují nežádoucímu proudu, který by mohl téci do zdrojů. Připojení externího napájení je také indikováno LED D20. Napájecí větev 3,3V, kterou vyžadují některé obvody na desce je vyvedena z desky MBED, která obsahuje stabilizační prvek na napětí 3,3V.

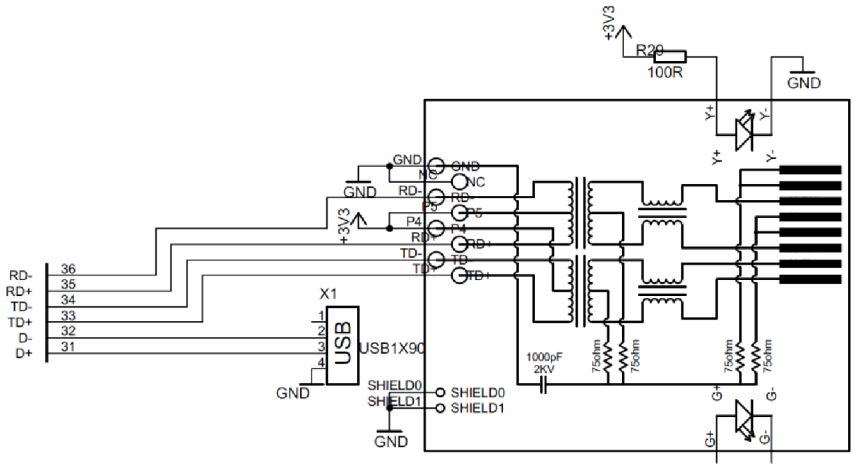


Obr. 17 Napájecí obvod Desky periferií

5.8.2 Připojení USB a Ethernet

Tato rozhraní slouží zejména pro komunikaci desky s PC. USB port je vyveden na konektor USB typu A-samice. Ethernet na standardní konektor RJ45. Pomocí USB lze

po připojení k PC a nainstalování odpovídajících ovladačů velice snadno vytvořit virtuální COM port. Ten můžeme využít v řadě laboratorních úloh pro jeho jednoduchost. Právě použití virtuálního COM portu je ukázáno v jedné laboratorní úloze níže. Ethernet bylo nutné připojit přes magnetický oddělovací transformátor, který je integrován přímo v konektoru RJ45. Žlutá kontrolní LED na konektoru na konektoru indikuje aktivitu rozhraní.



Obr. 18 Připojení USB a Ethernet

5.8.3 Návrh desky plošného spoje

Při návrhu DPS bylo nezbytné dodržet několik požadavků. Deska musela být koncipována jako jednostranný plošný spoj s nízkou konstrukční třídou a s minimem drátových propojek, tak aby bylo možné desku snadno a levně vyrobit. Protichůdný požadavek byl na malé rozměry desky. Dále bylo nutné vést dráhy vodičů pro připojení vysokofrekvenčních rozhraní (USB, Ethernet) v minimální délce, aby nemohlo docházet k odrazům ve vedení.

Samotný návrh desky byl proveden v návrhovém systému Eagle 5.0.0. Podařilo se dodržet všechny výše uvedené požadavky, avšak počet nutných drátových propojek je značně velký. Rozmístění jednotlivých součástek bylo značně podmíněno minimální délkou vodičů a tím bylo docíleno zmenšení rozměrů desky na 155x100. Při návrhu byly zejména použity součástky klasické montáže pro jejich snadné pájení. Použití těchto součástek výrazně nezvětšovalo rozměry desky. V rozích desky byly vytvořeny 4 montážní otvory, ke kterým byly namontovány krátké distanční sloupy, aby mohla deska stabilně stát na stole. Celkový návrh desky je uveden v příloze. Na doprovodném CD lze také nalézt všechny nezbytné podklady pro výrobu DPS.

6 LABORATORNÍ ÚLOHY

Následující řádky této kapitoly zahrnují hlavní část práce. Jsou zde uvedeny tři laboratorní úlohy včetně zadání a návodu, podle kterého by měli studenti úspěšně vytvořit danou úlohu.

Navržené úlohy jsou koncipovány s ohledem na procvičení ovládání nejčastěji používaných periferií připojovaných k mikrokontrolérům. Studenti se seznámí se zobrazováním dat na displeji či znázornění stavu zařízení na LED diodách. Naučí se naprogramovat sériovou komunikaci s periferním obvodem, která je řešena na úrovni hardwarové či softwarové. Dále je studentům ukázána komunikace mikrokontroléru s nadřazeným systémem. Všechny úlohy mají i praktické využití v aplikacích, kde jsou nasazovány mikrokontroléry. Programováním zadaných úloh si taktéž studenti osvojí poznatky z programovacího jazyka C++.

Každá úloha je rozčleněna na tři části – Zadání, Teoretický úvod a Popis řešení. V první části je v několika logických bodech chronologicky zapsáno zadání úlohy. Je nutné, aby studenti vypracovali úlohu postupně po jednotlivých bodech zadání, protože jednotlivé body na sebe navazují a proto není možné jejich přeskakování. Většina bodů zadání také po jejich dokončení tvoří funkční část dané úlohy, takže pokud student nestihne vypracovat celou úlohu, může aspoň vyzkoušet část úlohy. Každá z těchto úloh vždy přináší ukázku použití nového periferního obvodu či nové programovací techniky. Právě tímto se zabývá část Teoretický úvod, ve které se mají studenti seznámit s hlavní problematikou dané úlohy. Vlastní implementace těchto poznatků je zapsána v poslední části Popis řešení. Zde je k většině bodů zadání popsán postup pro jeho úspěšné řešení. Některé body zadání zde nejsou popsány, jelikož mají jen seznamovací charakter. Na konci úlohy je vždy uveden vývojový diagram úlohy pro snazší pochopení daného algoritmu. Případně jsou uvedeny ukázky kódů, které mají většinou charakter šablony ve formě znázornění zápisu daného algoritmu v jazyce C++.

Náročnost jednotlivých úloh je stejná. K vyvážení jejich náročnosti napomáhá i vytvoření trojce knihoven k ovládání navržených obvodů LCD displeje, sedmi-segmentového displeje a tlačítek. Tyto knihovny jsou popsány v příloze. Pro některé obvody si ovšem musí studenti vytvořit knihovnu sami v rámci řešení úlohy. Knihovny pro ovládání periferií desky MBED jsou k dispozici na webu výrobce [4] i s detailním popisem.

V poslední části řešení úlohy by se měli studenti snažit program patřičně odladit. Zde nastává problém, jelikož deska MBED neumožňuje jakoukoliv možnost ladit program za běhu. Řešením může být použití simulátoru ve vývojovém prostředí uVision. Případně využít čtveřici LED diod na desce k interpretaci stavu programu.

6.1 Digitální hodiny využívající obvod reálného času (RTC)

6.1.1 Zadání

1. Seznamte se s obvodem RTC integrovaném na vývojové desce MBED. Prostudujte knihovnu sloužící k ovládání tohoto obvodu.
2. V jazyce C++ vytvořte třídu reprezentující digitální hodiny, které budou obsahovat informace o aktuálním čase, roce, měsíci v daném roce a dni v měsíci. Výchozí čas prozatím nastavte na čas 00:00, rok 2000, den 1.1. Tato třída bude využívat obvod RTC. Jako zobrazovací zařízení zvolte LCD ATM0802A a pro jeho ovládání použijte příslušnou knihovnu. Třídu implementujte jako typ Singleton (jedináček), jelikož vývojová deska postrádá více jak jeden obvod RTC.
3. Třídu rozšiřte o možnost uživatelského nastavení aktuálního času pomocí jednoduchého menu. Jako vstupní zařízení použijte tři tlačítka.
4. Vytvořte objekt dané třídy a program odlaďte.

6.1.2 Teoretický úvod

6.1.2.1 Hodiny reálného času (RTC)

Hodiny reálného času jsou na desce MBED ve formě integrovaného obvodu. Tento obvod udržuje velmi přesnou informaci o aktuálním čase nezávisle na běhu procesoru a s využitím minima elektrické energie. Napájení bývá velice často řešeno pomocí malé knoflíkové baterie, pomocí které je obvod nepřetržitě pod napětím a tím je možný běh obvodu RTC i po odpojení hlavního napájení. Tato baterie však na této desce nebyla použita, proto je nutné po každém zapnutí obvod RTC inicializovat. Poté je možné kdykoliv načíst informaci o aktuálním čase.

K ovládání obvodu RTC slouží knihovny *RTC_time.h* a *time.h*. Tyto knihovny poskytují základní funkce pro inicializaci, nastavení výchozího času a také zpětné načtení aktuálního času z obvodu. Následující řádky uvádějí nejdůležitější funkce pro ovládání tohoto obvodu.

```
void set_time(time_t)
```

Nastavení výchozího času pro RTC obvod. Parametr *time_t* udává počet sekund od data 1. 1. 1970.

```
struct tm* localtime(const time_t*)
```

Provádí vytvoření struktury *tm* na základně počtu sekund.

```
time_t time(time_t*)
```

Tato funkce vrátí informaci o aktuálním čase je formě počtu sekund od data 1. 1. 1970. Jako parametr je nutné zadat prázdný ukazatel (***NULL***).

```
time_t mktime(struct tm*)
```

Vypočítá počet sekund od 1. 1. 1970 na základě struktury *tm*.

Výše uváděná struktura *tm* obsahuje jednotlivé informace o počtu hodin, minut, dnu v týdnu atd. Následující ukázka kódu demonstruje inicializaci i zpětné načtení času z obvodu RTC.

```
//-------------------------------------------------------------
//Nastavení výchozího času
struct tm stime = {0,0,12,1,1,100}; //Nastavení na 12:00 1.1.2000
time_t seconds = mktime(&stime);          //Vypočet sekund
set_time(second);                   //Nastaví RTC

//-------------------------------------------------------------
//Načtení času z RTC
seconds = time(NULL);               //Načtení z RTC
tm *t_ptr = localtime(&seconds);    //Výpočet struktury
myLCD.printf(„%d:%d",t_ptr->tm_hour,t_ptr->tm_min);//Výpis času
```

6.1.2.2 Návrhový vzor Singleton

Tento návrhový vzor se používá zejména tam, kde je potřeba vytvořit pouze jednu instanci dané třídy. Tímto zabezpečíme pouze jeden globální přístupový bod k jedné instanci a tím znemožnění vytvoření dalších instancí. Výše uvedené řešení bude obsahovat třídu reprezentující digitální hodiny. Právě pro takovou třídu je použití tohoto návrhového doporučené, jelikož nelze vytvořit více instancí této třídy z důvodu pouze jediného obvodu RTC.

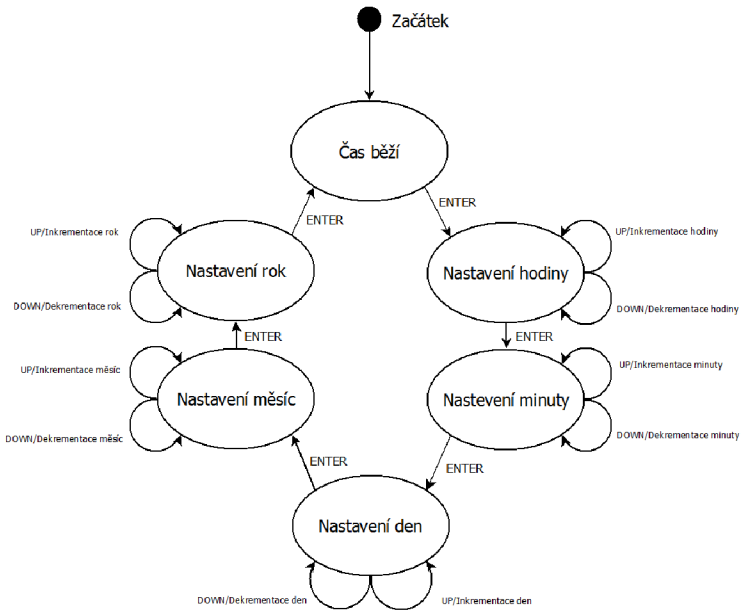
Implementaci v jazyce C++ lze provést přidáním statického ukazatele na danou třídu a také statické metody do navrhované třídy. Při prvním volání statické metody dojde k dynamické alokaci dané třídy, jejíž ukazatel se uloží do statické proměnné. K instanci pak lze přistupovat pouze pomocí tohoto statického ukazatele. Následující kód nastiňuje šablonu pro vytvoření tohoto návrhového vzoru.

```
class Singleton {        //třída podle vzoru Singleton
private:
    static Singleton *instance;      //globální přístupoví bod
public:
    static Singleton *GetInstance() {      //vytvoření instance
        if (instance == NULL) {      //vytvoří se pouze jednou
            instance = new Singleton();    //alokace nové instance
        }
        return instance;
    }
}
//-------------------------------------------------------------
int main(void)
{
 Singleton::GetInstance();     //Vyvoření instance
}
```

6.1.3 Popis řešení

ad 2.) Nejprve vytvoříme třídu *LCD_clock*, která bude reprezentovat naše digitální hodiny. Do této třídy můžeme hned implementovat výše zmíněný návrhový vzor Singleton. Dále v konstruktoru třídy nastavíme výchozí čas pro obvod RTC podle zadání na 00:00, rok 2000, den 1.1. Pro zobrazení času na displeji je nutné periodicky načítat čas z obvodu RTC a provést jeho výpis na displej minimálně jednou za sekundu. K tomuto účelu nám nejlépe poslouží časovač, jehož ovládací funkce najdeme v knihovně *ticker.h*. Časovač je zde zapsán ve formě třídy *Ticker*. Důležitými metodami této třídy časovače jsou *attach* a *detach*. Pomocí *attach* můžeme spustit časovač, který bude periodicky volat námi zadanou metodu. Jako parametr metody *attach* musíte zadat ukazatel na tuto metodu a také interval volání této metody. Tato periodicky volaná metoda je zobrazena ve vývojovém diagramu na Obr. 20 v rámečku. Před tímto je však nutné přidat do naší základní digitálních hodin ukazatel na LCD, jehož adresa bude vložena v konstruktoru jako parametr. Poté na displej zapisovat.

ad 3.) Pro uživatelské nastavení času je nejprve nutné přidání tří ukazatelů do třídy *LCD_clock* na třídu *push_button* přestavující jedno tlačítko. Ukazatelé na tlačítko se budou opět předávat parametry konstruktoru třídy *LCD_clock*. Samotné uživatelské nastavení je vhodné řešit na principu Mealyho stavového automatu, který je zobrazen na Obr. 19.



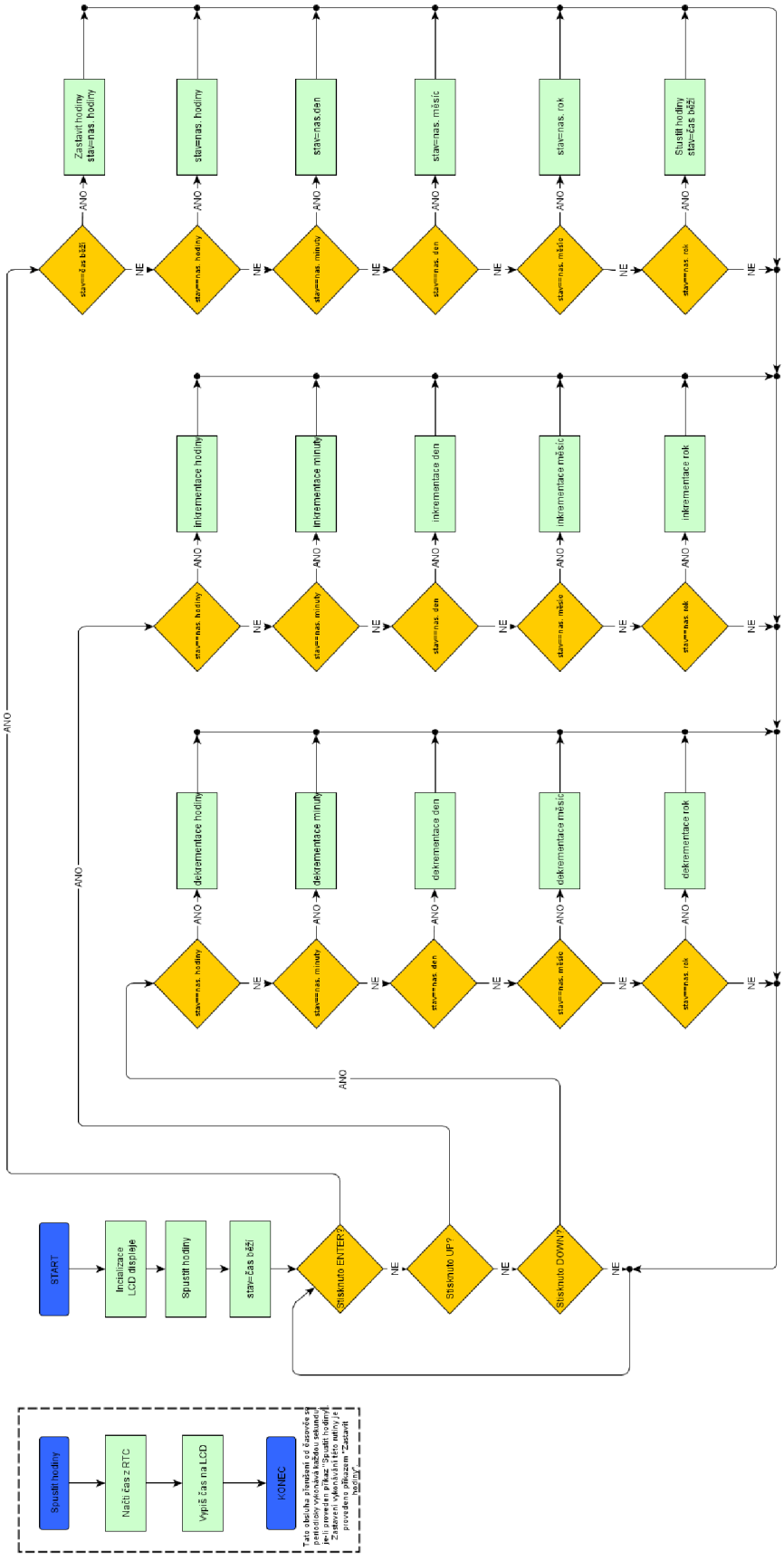
Obr. 19 Stavový diagram digitálních hodin

Tento kruhový stavový diagram obsahuje šest stavů, z něhož pět slouží k uživatelskému nastavení a jeden vystihuje stav, kdy jsou hodiny spuštěny. Po spuštění programu přejdou digitální hodiny do stavu *Čas běží*. Chce-li uživatel nastavit nový čas zmáčkne tlačítko *ENTER* a tím dojde ke změně stavu na stav *Nastavení hodiny*. Nyní

pomocí tlačítek *UP* a *DOWN* může uživatel nastavit novou hodnotu a dalším stiskem *ENTER* přejít na nastavení minut atd. Tímto způsobem lze jednoduše nastavit všechny hodnoty a dostat se zpět do stavu *Čas běží.* Pro programovou implementaci takového stavového diagramu do jazyka C++ je vhodné do naší základní třídy *LCD_clock* digitálních hodin přidat stavovou proměnou, která bude přestavovat jeden ze šesti výše uvedených stavů. Program poté může v hlavní smyčce neustále testovat hodnotu stavové proměnné a také stavy tlačítek a podle těchto informací provádět dané operace. K tomu to účelu použijeme větvení programu podle následující šablony.

```
for(;;){      //hlavní smyčka
 if(!enter->read()) {          //Je stisknuto tlačítko ENTER?
       switch(state)           //Zjištění stavu.
       {
             case run:   {/*Požadované příkazy */ break;}
             case set_hour:{state=set_min;break;}
             case set_min: {state=set_day;break;}
             case set_day: {state=set_mon;break;}
             case set_mon: {state=set_year;break;}
             case set_year:{/*Požadované příkazy*/break;}
       }
 }
//Testování dalších tlačítek.
}
```

V hlavní smyčce se testuje, zda je stisknuto některé ze tří tlačítek. Dojde-li ke stisku tlačítka *ENTER* program vyhodnotí podle stavové proměnné stav zařízení a provede požadované operace dle stavového diagramu. Obdobným způsobem provedeme zápis zdrojového kódu i pro tlačítka *UP* a *DOWN*.



Obr. 20 Vývojový diagram – Digitální hodiny využívající obvod reálného času

6.2 Měření vstupního napětí pomocí AD převodníku.

6.2.1 Zadání

1. Na pin p16 kanálu AD převodníku desky MBED připojte potenciometr, tak aby bylo možné pomocí něj nastavovat vstupní napětí AD převodníku. Hodnotu potenciometru zvolte minimálně 10 kΩ.
2. Použitím knihovny *analogin.h* načtete vstupní napětí připojeného kanálu a hodnotu převeďte do tvaru ve voltech.
3. Hodnotu vstupního napětí převeďte na dvojici cifer, které se budou zobrazovat na dvojici sedmi-segmentových displejů. První cifra z leva na displeji bude uvádět jednotky a druhá desetiny voltů.
4. Realizujte sériovou komunikaci s ovladačem LED diod TLC5940.
5. Změřenou hodnotu z AD převodníku grafiky interpretujte pomocí šestnácti LED diod. Ze šestnácti LED diod vytvořte bar graf, v němž počet rozsvícených diod na plný jas bude odpovídat hodnotě vstupního napětí dělenou konstantou 0,20625 (3,3/16=0,20625) a jas následující diody bude úměrný zbytku po dělení touto konstantou. Například: Vstupní napětí 2,9V. Počet rozsvícených diod na jas 100% je čtrnáct. Patnáctá dioda je rozsvícena na 6%.

6.2.2 Teoretický úvod

6.2.2.1 Integrovaný AD převodník

Mikrokontrolér LPC1768 má v sobě integrovaný AD převodník pro měření analogových hodnot. Tento převodník pracuje na principu postupné aproximace s rozlišením 12 bitů a s maximální pracovní frekvencí 200 kHz. Referenční napětí pro převodník je pevně nastaveno na hodnotu 3,3V a jeho změna z pohledu programátora není možná. Spodní hranice je definovaná analogovou nulou, jejíž hodnota je 0V. Převodník je připojen na výstup desky přes osmi-kanálový analogový multiplexor.

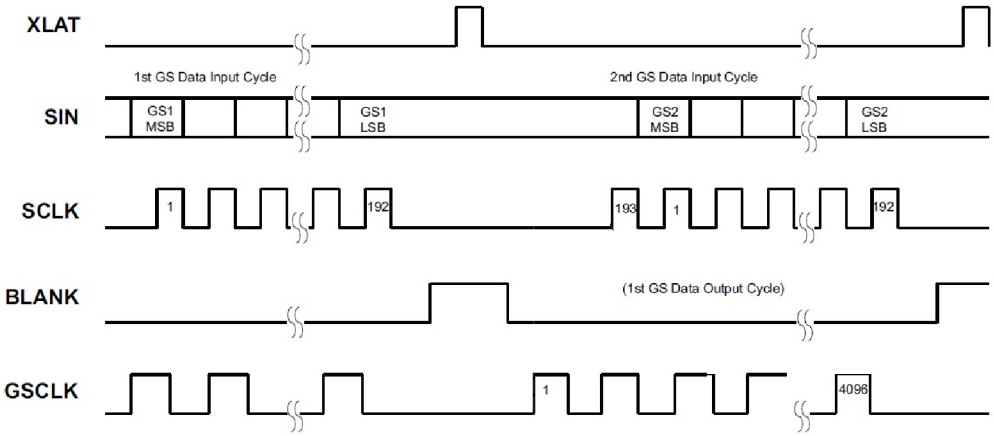
Ovládání převodníku je velmi snadné. K našim účelům poslouží knihovna *analogin.h*. Zde je možné najít požadovanou třídu *AnalogIn*, která je schopná zajistit veškerou komunikaci a nastavení obvodu. Třída obsahuje pouze jeden konstruktor a dvě metody pro řízení celého obvodu. Při vytváření objektu této třídy vložíme do konstruktoru název pinu, ze kterého máme v úmyslu měřit analogový signál. Tento pin však musí v rozsahu p15 až p20, protože pouze tyto vývody jsou zapojeny na vstup multiplexoru. Chceme-li změřit napětí na nastaveném pinu, použijeme metodu *read*. Tato metoda vrátí hodnotu v datovém typu float v rozsahu 0.0f až 1.0f. Pokud požadujeme převedení na hodnotu uvedenou přímo ve voltech, postačí vynásobení konstantou referenčního napětí 3,3V jak již bylo uvedeno výše. Další možnosti jak číst data je zavolání metody *read_16,* ze které získáme proměnou typu int v rozsahu 0x0 až 0xFFFF.

6.2.2.2 Obvod TLC5940

Tento ovladač LED diod slouží k řízení až 16 diod prostřednictvím sériového rozhraní. Je možná také regulace jasu jednotlivých LED s využitím PWM. Obvod je připojen k desce MBED prostřednictvím 5 signálů podle Tab. 9, kde uveden název pinu a také k jakému pinu Desky periferií byl připojen. Zbývající vstupy VPROG a DCPRG jsou připojeny na konstantní logické úrovně a pro naše účely nejsou důležité. Vnitřní zapojení obvodu v sobě obsahuje 16 registrů GS, v níž je uložena hodnota odpovídají svítivosti jednotlivých LED. Dále můžeme uvnitř obvodu nalézt 16 DC registrů, které slouží pro korekce svítivosti jednotlivých LED diod. V našem zapojení však tyto registry nelze používat, jelikož k nim nejsou zapojení potřebné vstupní piny. Celý obvod pracuje následovně. Nejdříve je nutné držet signál XLAT v úrovni L. Následně je do obvodu odesláno 192 bitů (16 GS registrů po 12 bitech), které se uloží do vstupního posuvného registru. Při ukončení odeslání posledního bitu jsou data potvrzena kladným impulzem XLAT a tím dojde k překopírování dat ze vstupního posuvného registru do odpovídajících GS registrů. Aby mohl obvod správně pracovat, je nutné na vstup GSCLK neustále přivádět hodinový signál pro generování PWM signálu. Tento signál inkrementuje GS čítač a hodnota v tom čítači je neustále porovnávána s hodnotou uloženou v GS registru. Jsou-li hodnoty shodné (GS čítač=GS registr) dojde odpojení odpovídající LED diody. Jakmile GS čítač dosáhne svojí maximální hodnota (4095) je nezbytné přivedení signálu BLANK, s jehož pomocí dojde k resetování GS čítače a opětovné připojení LED diod. Tímto způsobem lze na jednotlivých výstupech generovat PWM signál. Celkový průběh zápisu dat do obvodu je zobrazen na Obr. 21.

Název pinu	Připojeno na	Popis funkce
SIN	p5	Vstup pro sériová data.
SCLK	p6	Hodinový signál pro vstupní data.
XLAT	p7	Při kladném impulzu XTAL dojde k překopírování dat ze vstupních posuvných registrů do GS registrů.
BLANK	p8	Je-li BLANK=H jsou odpojeny všechny výstupy a je resetován GS čítač. Při BLANK=L jsou výstupu ovládány pomocí PWM.
GSCLK	p28	Hodinový signál pro generování PWM signálu.

Tab. 9 Popis připojení pinů obvodu TLC5940 na Desku periferií



Obr. 21 Časový diagram zápisu do obvodu TLC5940 (převzato a upraveno z [8])

6.2.3 Popis řešení

ad 1.) Na pin p16 připojíme jezdce potenciometru. Další dva vývody potenciometru zapojíme na zem (GND) a na napájecí napětí 3,3V. Napětí 3,3V můžeme získat i z vedlejšího vstupně-výstupního pinu p15 jeho nastavením do úrovně H.

ad 2.) Pro použití AD převodníku přidáme knihovnu *analogin.h* do nového projektu. Vytvoříme novou instanci třídy *AnalogIn* pro pin p16. Použitím metody *read* načteme vstupní hodnotu a tu vynásobíme konstantou 3,3 pro získání hodnoty ve voltech.

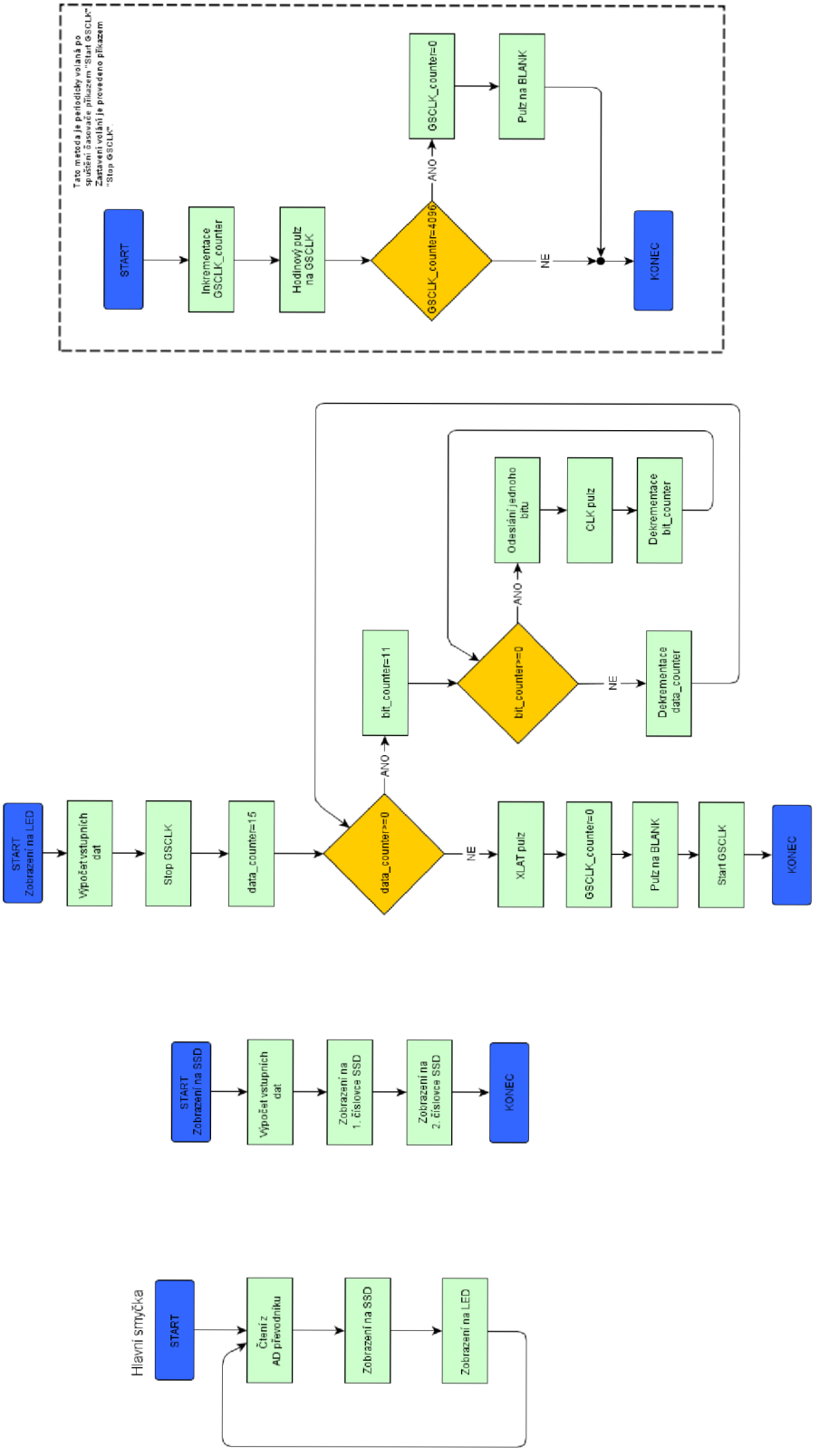
ad 3.) Přidáme do projektu další knihovnu *SSD.h*. Tato knihovna nám zajistí zobrazování dat na sedmi-segmentovém displeji. Vytvoříme dva objekty třídy SSD, kde jeden tento objekt přestavuje jednu číslovku sedmi-segmentového displeje. Při vytváření těchto objektů vložíme do konstruktorů požadované piny pro připojení displejů. Nyní musíme načtenou hodnotu z AD převodníku převést na dvojici číslovek pro první a druhou pozici displeje. K tomu účelu se nejlépe chodí operátory násobení a modulo (zbytek po dělení).

ad 4.) K ovládání obvodu TLC5940 vytvoříme novou třídu pod názvem *TLC5940*, jenž nám zajistí veškerou komunikaci s obvodem. Aby mohl obvod na svých výstupech generovat PWM signál je nutné neustále na jeho vstup GSCLK přivádět hodinový signál. Na generování tohoto signálu je možno použít časovač. Časovač najdeme jako třídu *Ticker* v knihovně *ticker.h*. Jeho přidáním budeme schopni periodicky volat námi zadanou metodu *gs_clk*. V této metodě vyrobíme jeden hodinový pulz přivedený na vstup GSCLK. Délka trvání hodinového pulzu není kritická. Je možné použít okamžitou změna z úrovně L na H a zpět jakékoliv vyčkávací rutiny. Dále je nezbytné počítat počet vyslaných pulzů. K tomu nám poslouží nová proměnná *GSCLK_counter*, kterou budeme v po každém cyklu inkrementovat. Jakmile program napočítá 4096 vyslaných impulzů, je nutné *GSCLK_counter* vynulovat a odeslat na vstup BLANK kladný pulz.

Nyní postačí vytvořit metody *start_gsclk* a *stop_gsclk,* ve kterých budeme zapínat či vypínat časovač.

Chceme-li odesílat data do obvodu, musíme vytvořit novou metodu *send_data*. Jako parametr této metody zvolíme pole šestnácti číselných hodnot typu unsigned int. Jedno číslo v tomto poli bude představovat svítivost jedné LED diody v rozsahu 0 až 4095. K samotnému odesílání dat je nejvhodnější používat dvojici vnořených cyklů typu for. Ve vnější smyčce budeme procházet jednotlivé hodnoty v poli. Ve vnitřní smyčce je pak nutné procházení jednotlivých bitů. Po vystavení jednoho bitu na výstup je poté nutné jeho potvrzení hodinovým signálem *CLK*. Po skončení obou smyček je nutné vyslat kladný impulz *XTAL*, který zabezpečí přesun dat do požadovaných GS registrů.

Ad 5.) Vstupní pole dat pro metodu *send_data* vypočteme následovně. V první fázi naplníme hodnotou 0xFFF prvních *N* hodnot pole. Kde *N* získáme dělením vstupního napětí konstantou 0,20625 a poté konverzí na datový dat int. Naplnění pole provedeme ve smyčce for. V další fázi vypočteme rozdíl mezi hodnotu před konverzí a po konverzi na typ int a tím získáme hodnotu v rozsahu 0 až 1 typu float. Takto získanou hodnotu vynásobíme konstantou 4095, konvertujeme ji na typ int a zapíšeme jako následující hodnotu pole. Všechny ostatní hodnoty pole musí být nulové, což zajistíme v počáteční inicializaci pole.



Obr. 22 Vývojový diagram – Měření vstupního napětí pomocí AD převodníku

6.3 Digitální teploměr s historií naměřených dat

6.3.1 Zadání

1. Realizujte komunikaci s teplotním čidlem LM92. Načtěte z čidla informaci o aktuální teplotě.
2. Pomocí časovače zobrazujte s periodou 1s načtenou teplotu na obrazovce LCD displeje.
3. Vytvořte dvourozměrné statické pole 360x2, do kterého se bude v 10 sekundových intervalech zapisovat historie o aktuální teplotě a relativním čase od začátku měření. Po zaplnění celého pole bude docházet k přepisu počátečních hodnot.
4. Pomocí tlačítek umožněte uživateli procházení pole se zobrazením na displeji.
5. Doplňte program o možnost odesílání naměřených dat do PC přes virtuální COM port.

6.3.2 Teoretický úvod

6.3.2.1 Komunikace s LM92 přes I2C

Komunikace s teplotním čidlem LM92 je možná pouze pomocí rozhraní I2C. S použitím této poloduplexní sériové sběrnice můžeme z čidla načítat informaci o aktuální teplotě a také nastavovat dolní a horní teplotní meze, které vyvolávají přerušení.

Knihovna pro ovládání rozhraní I2C se nachází v souboru *i2c.h*. V této knihovně můžeme nalézt třídu *I2C,* ve které najdeme základní metody pro komunikaci po sběrnici. Pro naše účely budou nejdůležitější metody *start*, *stop*, *read* a *write*. S jejich použitím jsme schopni odstartovat komunikaci, zastavit komunikaci a také číst a zapisovat data či příkazy do obvodu. Obvod LM92 obsahuje šest standardních registrů a jeden speciální, ve kterém se uložen ukazatel na registr se kterým se momentálně pracuje. Celou komunikaci lze rozdělit do následujících kroků.

1. Zahájíme komunikaci příkazem *start*.
2. Použitím příkazu *write* vyšleme na sběrnici jeden bajt, kterým vybereme obvod, se kterým máme v úmyslu komunikovat. Je nutné do horních 7 bitů zadat adresu zařízení (v našem případě obvod LM92 má nadefinovanou adresu 48h). Spodní jeden bit určuje, zda se bude číst (1) či zapisovat (0). Vybereme možnost pro zápis.
3. Dalším příkazem *write* zapíšeme do obvodu hodnotu 0, čímž nastavíme ukazatel na registr, v čemž je uložena hodnota aktuální teploty.
4. Provedeme opětovné vyslání příkazu *start*. Není zahájíme novou sekvenci pro čtení z obvodu.
5. Příkazem *write* opět vyšleme bajt, v čemž horní 7 bitů definuje adresu obvodu, ale tentokrát pro LSB bit zvolíme hodnotu 1, čili čtení.

6. Dvojitým použitím příkazu *read* načteme všech 16 bitů z Temterature Register, který obsahuje informaci o aktuální změřené teplotě.
7. Sekvenci ukončíme příkazem *stop*.

6.3.2.2 Komunikace s PC přes virtuální COM port

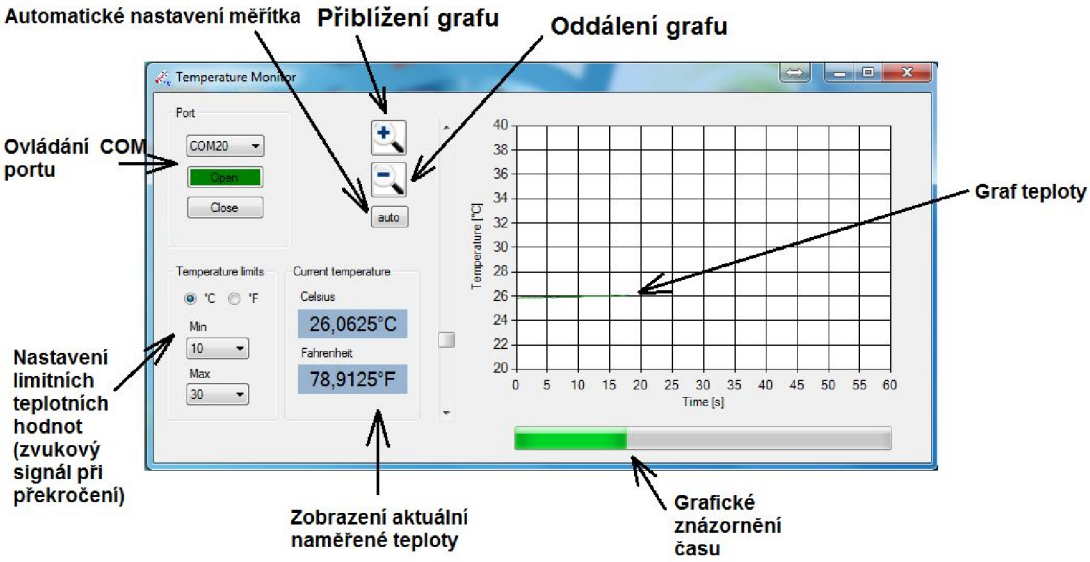
Importování knihovny *USBserial* do našeho projektu lze vytvořit virtuální COM port, pomocí kterého je velmi snadná komunikace s PC. Pro použití této knihovny je nezbytné propojení USB rozhraní desky MBED s PC. Výše uvedená knihovna v sobě obsahuje stejnojmennou třídu *USBserial*, která poskytuje řadu metod. Pro naše účely nám budou postačovat metody *printf*, *scanf* či *_getc* a *_putc*. Konstruktor této třídy můžeme použít s výchozími parametry. Zde je ještě nutné podotknout, že při vytváření instance této třídy je nezbytně nutné mít rozhraní USB desky MBED připojené k PC. Není-li tento kabel připojen, nedojde k pokračování běhu programu.

6.3.2.3 Temperature Monitor

Tento program byl vytvořen pro demonstraci komunikace přes virtuální COM rozhraní. Je napsán jako formulářová aplikace v prostředí MS Visual 2010 v jazyce C#. Zdrojový kód i s výsledným programem se nachází na doprovodném CD. Program umožňuje otevření vybraného COM portu. Po jeho otevření načítá data z portu, které musí být ve správném formátu. Je nutné, aby data, která reprezentují změřenou teplotu, byla přijímána periodicky každou sekundu. Přijímaný paket musí být ve formě textového řetězce. Tento řetězec vznikne z číselné proměnné, v níž je uložena hodnota aktuální teploty v °C, převedením pomocí standardní funkce *sprintf*. Například číselná proměnná typu double 22.214 musí být přijata ve formě textového řetězce o délce sedmi znaků obsahující znaky ‘2’,’2’,’.’,’2’,’1’,’4’,’\0’.

Ovládání tohoto programu je velmi snadné. V levé horní části se nachází rozhraní pro komunikaci. Zde je nutné vybrat příslušný virtuální COM port a provést připojení. V levé dolní části je dále možné vybrat teplotní limity, při jejíž překročení dojde ke spuštění varovných zvukových signálů. Celá pravá část je vyhrazena pro graf, který znázorňuje průběh vývoje teploty za poslední minutu. Vertikální osu tohoto grafu lze nastavit pomocí tlačítek nalevo od grafu. Tlačítkem auto můžeme zvolit automatické měřítko, které je odvozeno od změřených hodnot. Ve spodní části najdeme aktuální změřené hodnoty v jednotkách C i F. Ovládací rozhraní programu je popsáno na Obr. 23.

V neposlední řadě je nutné dodat, že pro správnou funkci programu je nezbytné nainstalovat patřičné ovladače pro virtuální COM port do PC, které jsou dostupné pro operační systémy Windows na stránkách [4] výrobce vývojové desky MBED.



Obr. 23 Popis ovládání programu Temperature Monitor

6.3.3 Popis řešení

ad 1.) Pro ovládání čidla LM92 vytvoříme novou třídu *LM92*, jenž bude obsahovat všechny potřebné metody pro ovládání obvodu. Základem naší nové třídy bude vložená třída *I2C* popsaná v kapitole 6.3.2.1. Pomocí této třídy načteme všech 16 bitů z Temperature registru obvodu LM92. Rozpis jednotlivých bitů tohoto registru je uveden na Obr. 24. Jak můžeme vidět, spodní 3 bity obsahují informaci o stavu obvodu a pro naše účely nebudou důležité. Horních 13 bitů nesou informaci o teplotě ve formátu dvojkového doplňku, kde LSB bit odpovídá teplotě 0,0625°C. Pro snadnou zpracovatelnost načtených dat je vhodné převodní do formátu double s využitím bitových operátorů jazyka C++. Převod na datový typ double je ukázán v následujícím kódu.

```
temp[0] = interface.read(1);
temp[1] = interface.read(0);          //načtení dat z obvodu
interface.stop();                     //zastavení komunikace
//převedení načtených dvou bajtů na číslo typu double
double temp_double = double(((temp[0]<<8)|temp[1])>>3))*0.0625;
```

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Sign	MSB	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	CRIT	HIGH	LOW
													Status Bits		

Obr. 24 Popis Temterature Register (převzato z [7])

ad 2.) K zobrazování změřených hodnot použijeme LCD displej ATM0802A. Vytvoříme úplně novou třídu *thermometer* reprezentující digitální teploměr, do které vložíme knihovní objekt výše uvedeného displeje. Pro zajištění periodického výpisu hodnoty po jedné sekundě je ideální použití časovače. Časovač najdeme v knihovně *ticker.h* ve formě stejnojmenné třídy. S využitím metod *attach* a *detach* této třídy

navolíme periodu jedna sekunda volání na novou metodu *show* a spustíme časovač. V rutině periodicky volané metody *show* provedeme načtení hodnoty z obvodu LM92 a jeho zpracování a výpis na LCD displej.

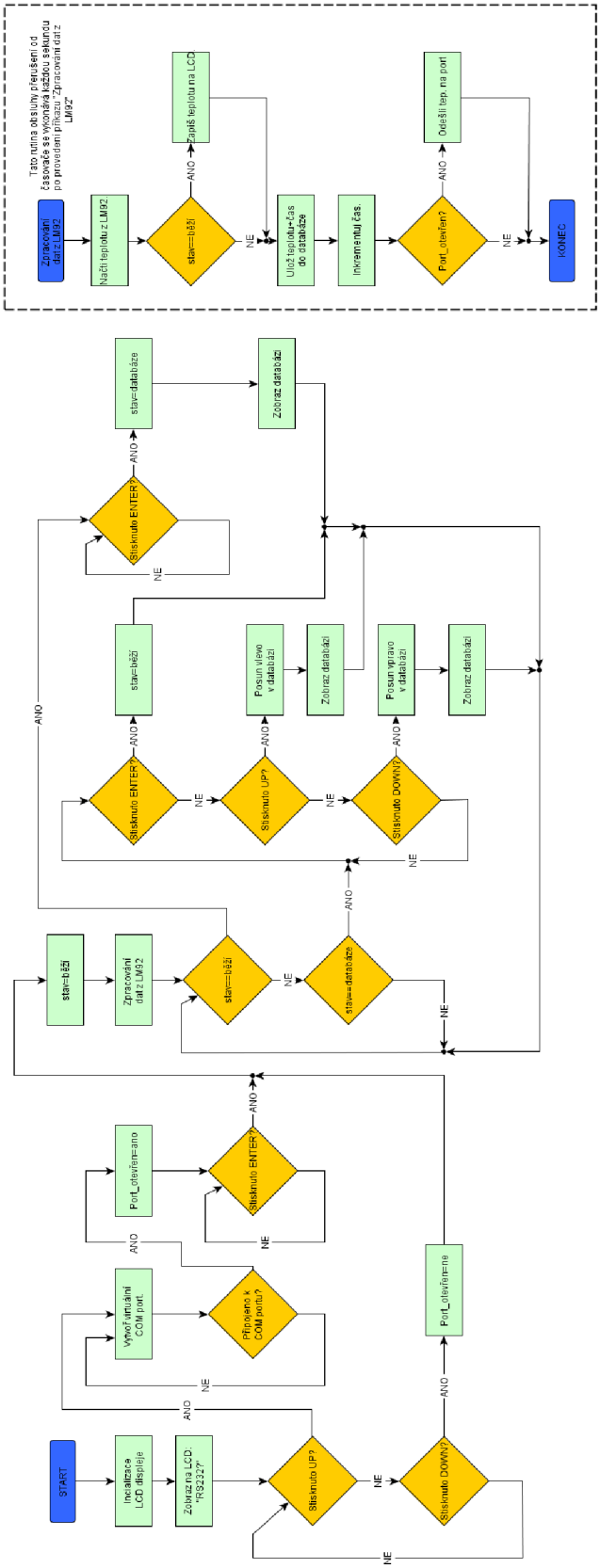
ad 3.) Pro splnění tohoto bodu je nezbytné vytvořit dvě nové třídy a to třídu *Ctime* a *Carray_temp*. Třída *Ctime* bude počítat čas od začátku měření po dobu jedné hodiny a po překročení této hodnoty dojde k vynulování času a opětovnému počítání od začátku. K inkrementaci času třídy *Ctime* můžeme použít periodicky volavou metodu *show* volanou pomocí časovače, která byla použita v předchozím bodu. Třída *Carray_temp* bude přetavovat dvourozměrné pole, kde jednou vstupní hodnou bude změřená hodnota a teplota a druhou hodnota čas ze třídy *Ctime*. Instance těchto dvou tříd vložíme do naší základní třídy *thermometer*. Nyní už stačí pouze v metodě *show* vládat hodnotu změřené teploty a čas to instance třídy *Carray_temp*.

ad 4.) Abychom splnili tento bod zadání, musíme uživateli umožnit přístup k datům v instanci třídy *Carray_temp*, v němž jsou obsažena všechna data o teplotě a času. Aby mohl uživatel program ovládat, je nutné přidat do naší základní třídy *thermometer* tři instance třídy *push_button*, které budou reprezentovat trojici připojených tlačítek. Dále přidáme do třídy stavovou proměnou, která nám bude určovat, zda program zobrazuje aktuální teplotu na displeji (stav *běží*) a nebo je v režimu databáze historie naměřených dat (stav *databáze*). Program v hlavní smyčce zjišťuje, ve kterém stavu se program momentálně nachází a poté testuje příslušná tlačítka. Je-li nějaké tlačítko stisknuto, provede se patřičná operace. Pokud se program nachází ve stavu *běží* a je stisknuto tlačítko *ENTER* přejde program do stavu *databáze*. V tomto stavu program testuje všechny tři tlačítka. Pomocí tlačítek *UP* a *DOWN* je možné procházení databáze historie naměřených dat. Opětovným stiskem tlačítka *ENTER* se program opět vrátí do stavu *běží*. Tuto část programu nejlépe vystihuje vývojový diagram na Obr. 25. V neposlední řadě je nezbytné vložit do periodicky volané metody *show* podmínku, aby docházelo k výpisu aktuální teploty pouze je-li program ve stavu *běží*.

ad 5.) Na začátek programu doplníme sekvenci, zda chce uživatel odesílat data do PC či nikoliv. Vytvoříme smyčku, ve které budeme testovat tlačítka *UP* a *DOWN* a také přidáme novou stavovou proměnou *port_otevren*, kterou budeme určovat, zda je port otevřen či ne. Bude-li stisknuto tlačítko *UP* program se pokusí dynamicky vytvořit instanci třídy *USBserial*. Zde je nutné dodat, že je nutné připojit USB kabel k PC jinak program objekt nevytvoří a bude vyčkávat, dokud nebude kabel připojen. V dalším kroku nastavíme novou stavovou proměnou *port_otevren* na *true*. Pokud stiskneme tlačítko *DOWN* program nastaví proměnou *port_otevren* na *false* a program bude dále standardně pokračovat. V této chvíli stačí to metody *show* vložit testování proměnné *port_otevren*. Nabývá-li proměnná hodnoty *true*, provede se odeslání na COM port. Data musí být ve formátu, jak bylo popsáno v 6.3.2.3, aby je mohl program

Temperature Monitor správně zpracovat. Pro odeslání na COM port použijeme metodu *printf*, která podobnou funkci jak stejnojmenná funkce ze standardní knihovny *stdio.h*, akorát místo výpisu do konzole provádí odeslání daného řetězce na port. Odeslání dat na port je uvedeno v následujícím kódu.

```
//Odesílání dat na virtuální COM port
if(port_otevren)         //Je port otevřen?
{
serial->printf("%f",temp_double);    //Odeslání ve formátu řetězce
}
```



Obr. 25 Vývojový diagram – Digitální teploměr s historií naměřených dat.

7 ZÁVĚR

V této práci jsem vytvořil podklady pro tři laboratorní úlohy k vývojové desce MBED NXP LPC1768. Tyto úlohy jsem koncipoval tak, aby se studenti zdokonalili v oblasti ovládaní periferií mikrokontrolérů a v programovacím jazyce C++.

Aby bylo možné úlohy realizovat, musel jsem navrhnout šest rozšiřující obvodů. Samotná deska MBED NXP LPC1768 totiž nabízí pouze čtveřici LED diod. Původní záměr realizovat tyto rozšiřující obvody na desce nepájivého pole se mě příliš neosvědčil. Sestavování jednotlivých obvodů bylo příliš zdlouhavé, neefektivní a také velmi náchylné na chybu při zapojování. Z těchto důvodů jsem se rozhodl vytvořit nad rámec zadání práce Desku periferií, která všechny navržené obvody sdružuje na jednu desku plošného spoje. Tuto desku jsem vytvořil jako jednostranný plošný spoj s drátovými propojkami na vrchní straně, aby její realizace nebyla příliš náročná a drahá. Dále jsem provedl připojení rozhraní USB a Ethernet pro možnost komunikace desky s okolím. U všech výše uvedených obvodů jsem odzkoušel jejich funkčnost. Výjimku tvoří obvod s připojeným maticové klávesnice, kde nebyla ověřena jeho funkčnost z důvodů špatné dostupnosti IO EDE1144. Maticová klávesnice však není v laboratorních úlohách použita. Dále jsem zjistil při testování desky zprvu nevhodné napájení 16ti LED diod, kdy při napájení jen z USB docházelo k překračování jeho maximálního dovoleného proudu. Proto jsem zvolil napájení LED diod pouze z externího zdroje. Tuto změnu jsem provedl na vyhotovené desce i poupravil výsledný návrh DPS.

Hlavní část práce tvoří zadaní a popis řešení tří laboratorních úloh - Digitální hodiny využívající hodiny reálného času (RTC), Měření vstupního napětí pomocí AD převodníku, Digitální teploměr s historií naměřených dat. Všechny úlohy jsem odladil na Desce periferií a ověřil jejich funkčnost. Všechny vytvořené úlohy i s komentářem zdrojového kódu jsou přiloženy v přílohách na CD.

Vytvořená Deska periferií nabízí celou řadu možností pro vytvoření dalších laboratorních úloh. Velmi zajímavé by bylo vytvoření úloh komunikující přes rozhraní Ethernet. Případně úlohy založené na čtení z klasické PC klávesnice či jiných zařízení, které je možno připojit přes rozhraní USB.

8 SEZNAM LITERATURY

[1] HRBÁČEK, Jiří. Komunikace mikrokontroléru s okolím 1. Praha: BEN technická literatura, 1999. 159 s. kód knihy 120921. ISBN 80-86056-42-2.

[2] HRBÁČEK, Jiří. *Komunikace mikrokontroléru s okolím 2.* Vyd. 1. Praha : BEN technická literatura, 2000. 151 s. kód knihy 120983. ISBN 80-86056-73-2.

[3] VÁŇA, Vladimír. ARM pro začátečníky. BEN - technická literatura, 2009. ISBN 978-80-7300-246-6.

[4] PAULGER, Stephena. Mbed NXP LPC1768. Mbed [online]. [cit. 2013-01-02]. Dostupné z: http://mbed.org/handbook/mbed-NXP-LPC1768

[5] LPC1769/68/67/66/65/64/63: 32-bit ARM Cortex-M3 microcontroller; up to 512 kB flash and 64 kB SRAM with Ethernet, USB 2.0 Host/Device/OTG, CAN. 2012, 82 s. LPC1769_68_67_66_65_64_63. Dostupné z: http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf

[6] DM74LS47: BCD to 7-Segment Decoder/Driver with Open-Collector Outputs. 2000. Dostupné z: http://pdf1.alldatasheet.com/datasheet-pdf/view/51080/FAIRCHILD/74LS47.html

[7] TEXAS INSTRUMENTS. LM92 ±0.33°C Accurate, 12-Bit + Sign Temperature Sensor and Thermal Window Comparator with Two-Wire Interface. 2010. Dostupné z: http://www.ti.com/lit/ds/symlink/lm92.pdf

[8] TEXAS INSTRUMENTS. 16 CHANNEL LED DRIVER WITH DOT CORRECTION AND GRAYSCALE PWM CONTROL. 2007. Dostupné z: http://www.ti.com/lit/ds/symlink/tlc5940.pdf

[9] TEXAS INSTRUMENTS. CD54HC4094, CD74HC4094, CD74HCT4094: High−Speed CMOS Logic 8−Stage Shift and Store Bus Register, Three−State. 2010. Dostupné z: http://www.ti.com/lit/ds/symlink/cd74hct4094.pdf

[10] E-LAB DIGITAL ENGINEERING, Inc. EDE1144 Keypad Encoder IC: 4 x 4 Matrix Keypad Encoder IC. Dostupné z: http://www.utm.edu/staff/leeb/encoder.pdf

[11] HITACHI. HD44780U (LCD-II): (Dot Matrix Liquid Crystal Display Controller/Driver). Dostupné z: http://pdf1.alldatasheet.com/datasheet-pdf/view/63673/HITACHI/HD44780.html

9 SEZNAM PŘÍLOH

I Popis knihoven pro práci s rozšiřujícími obvody 47
I.I Knihovna LCD_ATM0802A 47
I.II Knihovna SSD 48
I.III Knihovna Push_Button 49
II Deska periferií 50
I.I Fotodokumentace 50
I.II Schéma 52
I.III Deska plošného spoje 53
I.IV Seznam použitých součástek 53
III Obsah přiloženého CD 55

I Popis knihoven pro práci s rozšiřujícími obvody

I.I Knihovna LCD_ATM0802A

Tato knihovna zabezpečuje snadné a efektivní ovládání navrženého zapojení s LCD displejem, který byl popsán v kapitole 5.3. Knihovna byla implementována v jazyce C++ jako třída pod názvem LCD_ATM0802 obsahující veřejně přístupné metody a operátory.

Konstruktory

```
LCD_ATM0802A();
```

Implicitní konstruktor. Vytvoří nový objekt reprezentující objekt připojeného displeje. Výstupy pro připojení displeje jsou nastaveny na následující výchozí hodnoty *P18=Data*, *P19=Clock*, *P20=Strobe*.

```
LCD_ATM0802A(PinName _clk, PinName _dat, PinName _str, bool _backlight=false);
```

Konstruktor z trojce hodnot, které zastupují zvolené výstupu vývojové desky pro připojení displeje. Prostřednictvím proměnné _backlight je možné zapnout či vypnout podsvícení displeje.

Metody

```
virtual void printf(const char text[], ...);
```

Metoda pro standardní výstup na displej, která je analogií známé stejnojmenné funkce užívané v jazyce C, nacházející se v knihovně stdio.h pro tisk do konzole. První parametr je textový řetězec, který se má zobrazit na obrazovce. Obsahuje-li textový řetězec jeden z vyhrazených znaků, provede se daná speciální operace, která může vyžadovat další parametry. Prostřednictvím těchto znaků můžeme například vkládat do textu proměnné standardních typů, tabulátory či příkaz přesunu na další řádek. Tab. 10 ukazuje všechny vyhrazené znaky a jejich příklad použití.

Znak(y)	Popis	Příklad použití
%	Vložení hodnoty proměnné zadané v dalším parametru funkce, která vypíše na displeji ve formátu dle následujícího znaku.	Viz. následující příklady.

d	Následuje-li tento znak bezprostředně po znaku %, vypíše se na displeji proměnná ve formátu celého čísla. Za tímto znakem může následovat číslice 1 nebo 2, která určuje na kolik míst se má hodnota zobrazit. Lze tímto způsobem zobrazit případnou nulu zleva.	int A=15;
		myLCD.printf("A=%d", A);
		//Tento příkaz vypíše na displej:
		A = 1 5

f	Následuje-li tento znak bezprostředně po znaku %, vypíše se na displeji proměnná ve formátu desetinného čísla.	float A=15.5;
		myLCD.printf("A=%f", A);
		//Tento příkaz vypíše na displej:
		A = 1 5 . 5

/n	Skok na začátek druhého řádku displeje	myLCD.printf("Hello/nworld");

<table>
<tr><td></td><td></td><td>//Tento příkaz vypíše na displej:
<table><tr><td>H</td><td>e</td><td>l</td><td>l</td><td>o</td><td></td><td></td><td></td></tr><tr><td>w</td><td>o</td><td>r</td><td>l</td><td>d</td><td></td><td></td><td></td></tr></table></td></tr>
<tr><td>/t</td><td>Tabulátor – vynechání dvou znaků</td><td>myLCD.printf("Pocet/t5");
//Tento příkaz vypíše na displej:
<table><tr><td>P</td><td>o</td><td>c</td><td>e</td><td>t</td><td></td><td></td><td>5</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></td></tr>
</table>

Tab. 10 Popis speciálních znaků pro metodu printf

```
virtual void backlight(bool _backlight);
```

Metoda určená k ovládání podsvícení displeje. *_backlight*=TRUE – podsvícení zapnuto, *_backlight*=FALSE – podsvícení vypnuto.

```
void cursor(int positionX, int positionY, bool visibility=false);
```

Nastavení pozice kurzoru na displeji. Proměnná *positionX* nastavuje řádek (0-první řádek, 1-druhý řádek) a stejně tak *positonY* určuje sloupec, kde je rozsah 0-7. Parametrem *visibility* můžeme zvolit viditelnost kurzoru.

```
void erase(void);
```

Vymaže všechny znaky na displeji a nastaví kurzor na výchozí hodnotu 0,0.

```
void erase(bool line);
```

Provede vymazání zvoleného řádku a nastavení kurzoru na začátek daného řádku (0-první řádek, 1-druhý řádek).

```
void erase(int positionX, int positionY);
```

Na zvolené pozici realizuje vymazání znaku a přesun kurzoru na tuto danou pozici.

Operátory

```
LCD_ATM0802A& operator[](bool line);
```

Tento operátor indexování můžeme využít k nastavení pozice kurzoru na první či druhý řádek. Jde fakticky o jiný zápis metody *cursor*.

```
LCD_ATM0802A& operator<<(const char text[]);
```

Operátor standardního výstupu s využitím datových proudů. Použití je analogické s metodou printf, avšak zde nejsou definovány žádné vyhrazené znaky.

```
LCD_ATM0802A& operator<<(int value);
```

Pomocí tohoto operátoru lze provádět tisk proměnných typu int na obrazovku displeje.

```
LCD_ATM0802A& operator<<(char value);
```

Tisk jednoho znaku typu char na obrazovku.

I.II Knihovna SSD

Tato knihovna slouží k zobrazení číselných údajů na sedmi-segmentovém displeji. Implementace byla provedena pomocí třída, která reprezentuje jednu číslovku displeje.

Pro ovládání výše navrženého dvoumístného displeje je tedy nutné vytvořit dva objekty této třídy.

Konstruktory

```
SSD();
```

Implicitní konstruktor vytvoří objekt reprezentující jednu číslovku sedmi-segmentového displeje. Výstupy pro připojení budou implicitně nastaveny na *A=p20*, *B=p19*, *C=18*, *D=17*.

```
SSD(PinName A,PinName B,PinName C,PinName D,int number=0);
```

Konstruktor umožňující zvolit všechny výstupy pro připojení displeje. Parametr *number* určuje výchozí číslovku zobrazovanou na displeji.

Metody

```
void write(int number);
```

Umožňuje zápis číslovky na obrazovku v rozsahu hodnot 0-9. V případě vložení číslovky mimo tento rozsah se neprovede žádná operace.

```
inline int read(void);
```

Provede načtení zobrazované číslovky a její konverzi na hodnotu typu *int*.

Operátory

```
inline SSD& operator=(SSD& second);
```

Operátor přiřazení uskuteční zkopírování zobrazované číslovky, nicméně nedojde ke změně výstupních pinů pro připojení segmentu displeje.

```
inline SSD& operator=(int number);
```

Tento operátor má identickou funkci jako metoda *write*.

```
inline SSD& operator++(int);
```

Tímto operátorem můžeme inkrementovat zobrazovanou hodnotu. Dojde-li k přetečení nad hodnotu 9, tak se budou čísla zobrazovat od hodnoty 0.

```
inline SSD& operator--(int);
```

Zcela opačnou funkci oproti předcházejícímu operátoru má tento operátor. Jde o naopak o dekrementaci. Dojde-li k podtečení pod hodnotu 0, tak se budou čísla zobrazovat od hodnoty 9.

```
inline operator int();
```

Realizuje shodnou operaci jako metoda *read*;

I.III Knihovna Push_Button

Knihovna obsahující třídu reprezentující jedno tlačítko. Třída je schopná filtrovat zákmity tlačítka. Pro tento účet je však nutné zadat předpokládanou dobu trvaní zákmitu tlačítka.

Konstruktory

```
push_button(PinName pin, int Time=10);
```

Vytvoří nový objekt reprezentující připojené tlačítko. Parametr pin určuji pin pro připojení tlačítka. Parametrem *Time* se nastavuje předpokládaná doba trvání zákmitu tlačítka v milisekundách.

Metody

```
bool read(void);
```

Načte hodnotu z tlačítka. Tato metoda potlačuje případné zákmity tlačítka. *TRUE = připojeno na +Vcc*, *FALSE = připojeno na GND*.

Operátory

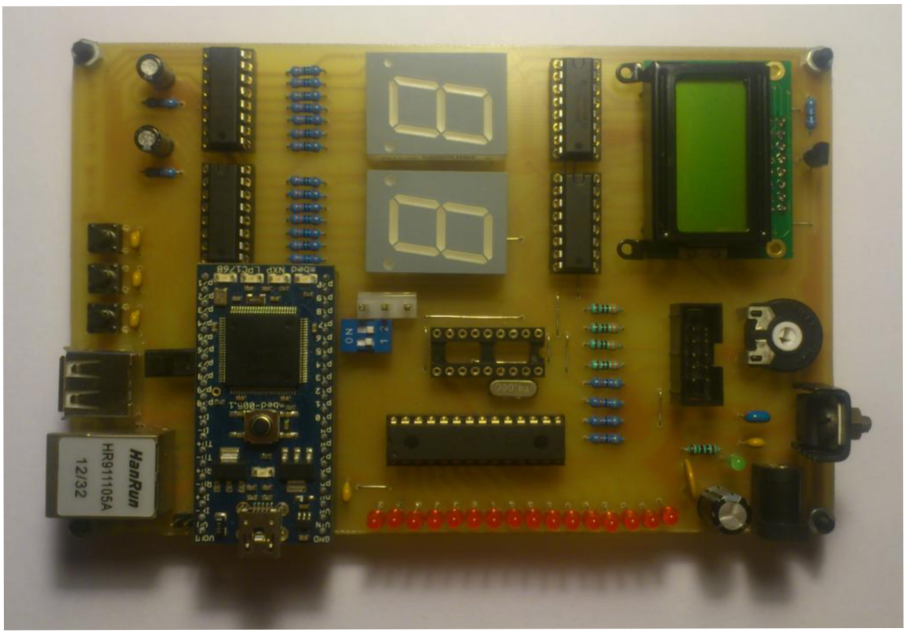
```
operator bool()
```

Ekvivalentní chování jako metoda *read*.

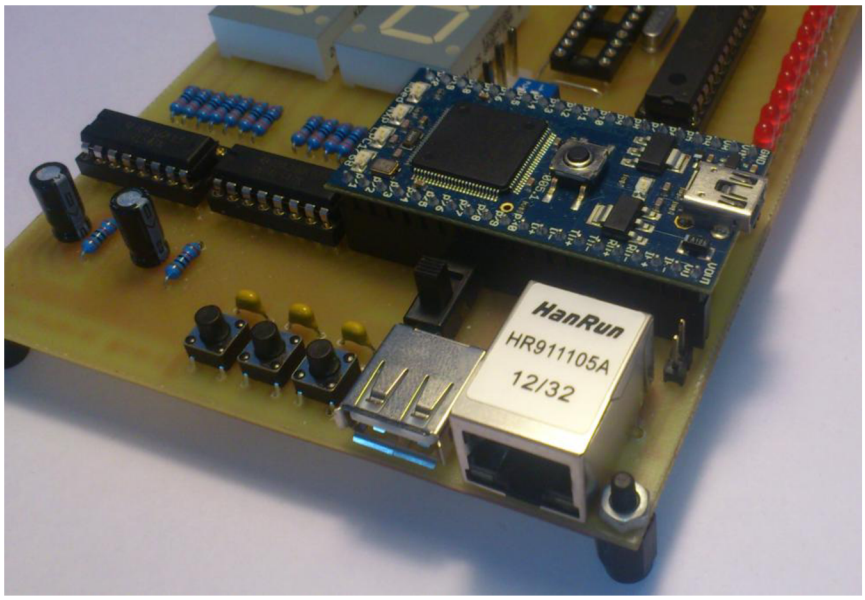
II Deska periferií

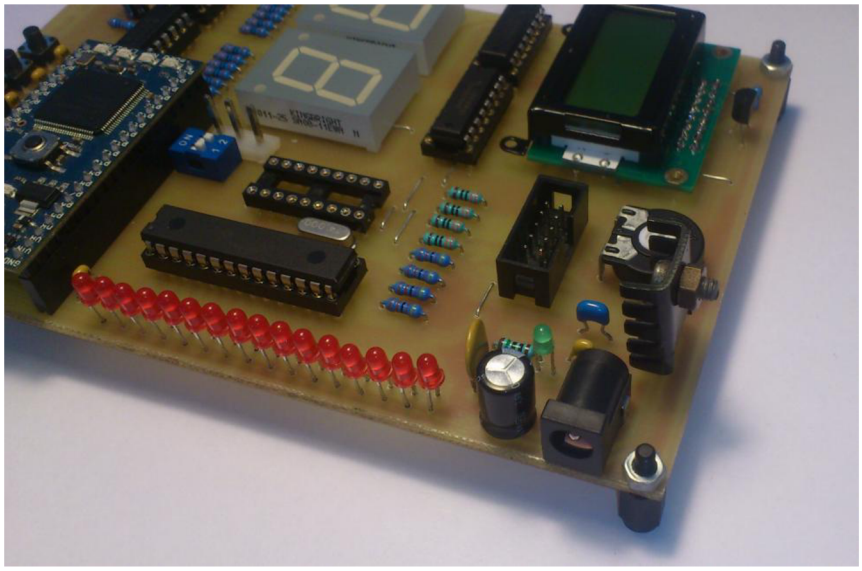
I.I Fotodokumentace

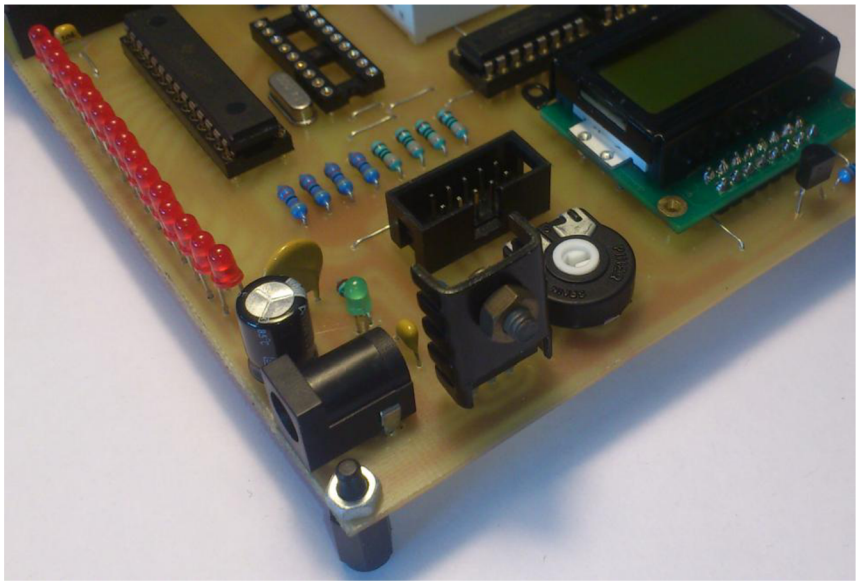
Celkový náhled na vyhotovenou desku



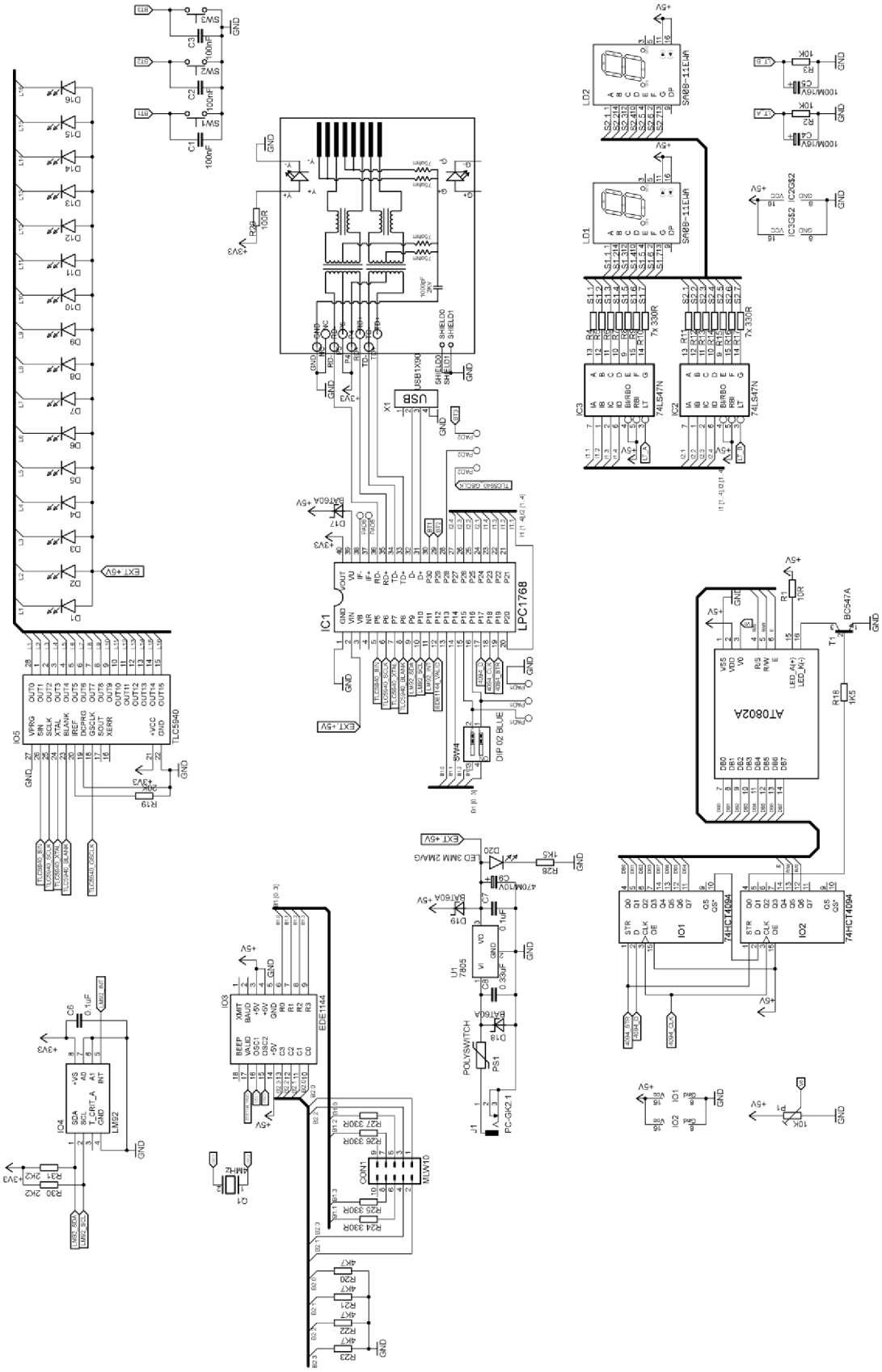
Detaily desky


HanRun
HR911105A
12/32



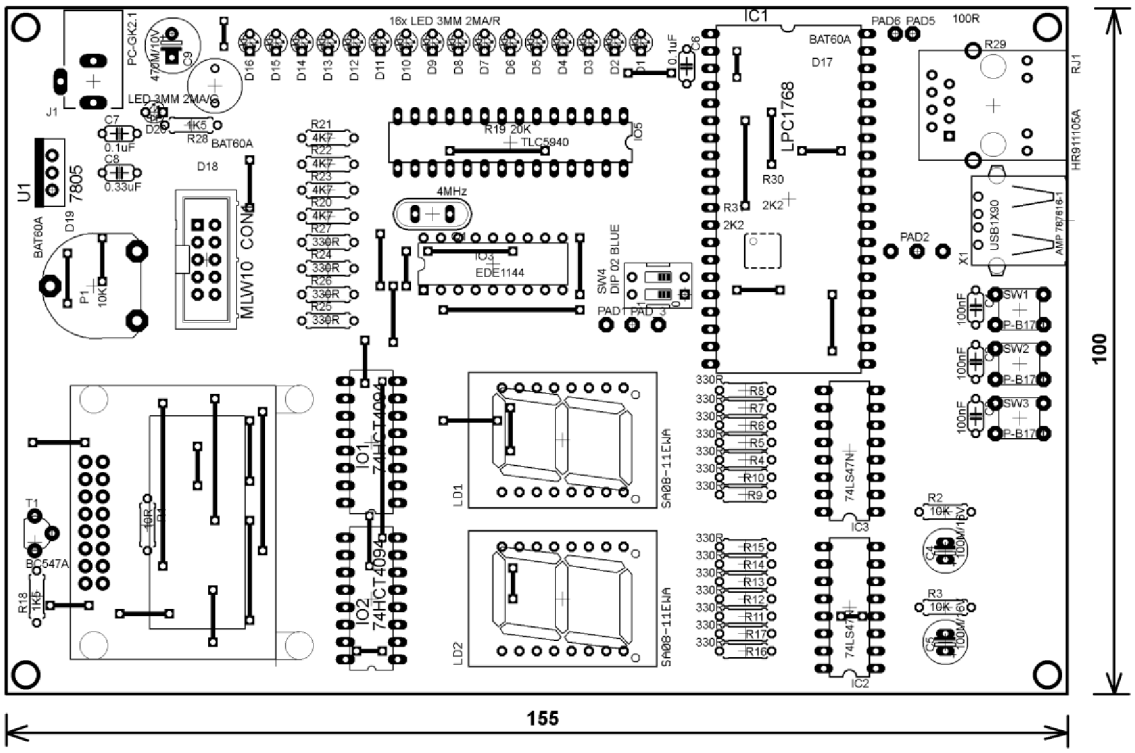


I.II Schéma

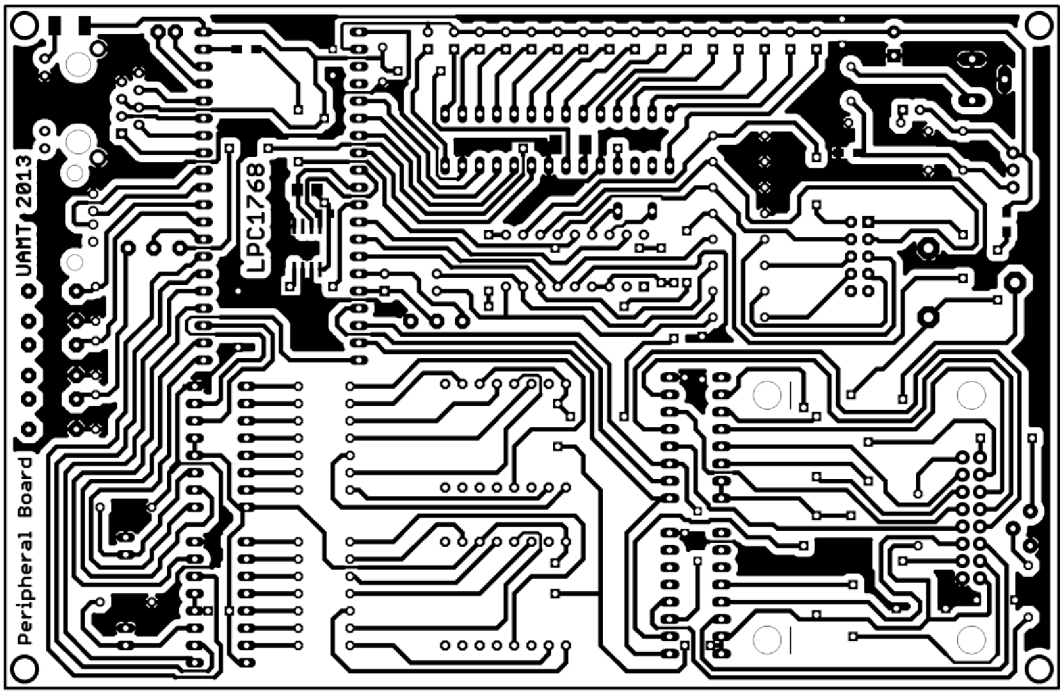


I.III Deska plošného spoje

Rozložení součástek ze strany Top (není v měřítku)



Plošný spoj ze strany Bottom (není v měřítku)



I.IV Seznam použitých součástek

C1-3	100nF
C4	100M/16V

C5	100M/16V
C6	0.1uF
C7	0.1uF
C8	0.33uF
C9	470M/10V
CON1	MLW10
D1-16	RED LED 3mm
D17-19	BAT60A
D20	LED 3MM 2MA/G LED
IC1	LPC1768
IC2	74LS47N
IC3	74LS47N
IO1	74HCT4094
IO2	74HCT4094
IO3	EDE1144
IO4	LM92
IO5	TLC5940
J1	PC-GK2.1
LD1	SA08-11EWA
LD2	SA08-11EWA
P1	10K PT15V
PS1	POLYSWITCH
Q1	4MHz QS
R1	10R 0207
R2	10K 0207
R3	10K 0207
R4-17	330R 0207
R18	1K5 0207
R19	20K R2010
R20-23	4K7 0207
R24-27	330R 0207
R28	1K5 0207
R29	100R R2010
R30	2K2 R1206
R31	2K2 R0603
RJ1	HR911105
SW1-3	P-B1720
SW4	DIP 02
T1	BC547A TO-92
U$2	AT0802A
U1	7805 TO-220
X1	USB1X90

III Obsah přiloženého CD

CD: \přílohy\Laboratorní úlohy\Vzorové vypracování laboratorních úloh

- Vypracované tři laboratorní úlohy

CD: \přílohy\Laboratorní úlohy\Knihovny pro laboratorní úlohy

- Knihovny k ovládání připojených periferií.

CD: \přílohy\Laboratorní úlohy\Temperature Monitor

- Program k laboratorní úloze „Digitální teploměr s historií naměřených dat“.

CD: \přílohy\Deska periferií

- DPS, rozmístění součástek na desce, 3D vizualizace desky, seznam součástek

CD: \přílohy\Deska periferií\Eagle soubory

- Soubory .sch a .brd návrhu desky vytvořené v systému Eagle.

CD: \přílohy\Literatura

- Dokumentace k použitým obvodům.

CD: \elektronická verze BP

- Kompletní bakalářská práce ve formátu pdf